User Manual

TextWrangler™

THE Text Editor for Anyone who Types

Bare Bones Software, Inc.

TextWrangler™ 5.0

Product Design	*Rich Siegel, Patrick Woolsey, Jim Correia, Steve Kalkwarf*
Product Engineering	*Jim Correia, Jon Hueras,Steve Kalkwarf, Rich Siegel, Steve Sisak*
Documentation	*Philip Borenstein, Stephen Chernicoff, John Gruber, Simon Jester, Jeff Mattson, Jerry Kindall, Caroline Rose, Rich Siegel, Patrick Woolsey*
Additional Engineering	*Seth Dillingham – Macrobyte Resources Polaschek Computing*
Icon Design	*Ultra Maroon Design* *updates by Bryan Bell*
Factory Text Color Schemes	*Luke Andrews*
PHP keyword lists	*contributed by Ted Stresen-Reuter http://www.tedmasterweb.com/*
Info-ZIP	*©1990-2009 Info-ZIP. Used under license.*
LibNcFTP	*©1996-2010 Mike Gleason & NcFTP Software*
NSTimer+Blocks	*©2011 Random Ideas, LLC. Used under license.*
PCRE Library Package	*written by Philip Hazel and ©1997-2004 University of Cambridge, England*
Quicksilver string ranking	*Adapted from available sources and used under Apache License 2.0 terms.*

TextWrangler and the TextWrangler User Manual are copyright ©2003-2015 Bare Bones Software, Inc. All rights reserved. Produced/published in USA.

Bare Bones Software, Inc.
73 Princeton Street, Suite 206
North Chelmsford, MA 01863 USA

(978) 251-0500 main
(978) 251-0525 fax

http://www.barebones.com/

Sales & customer service: **sales@barebones.com**
Technical support: **support@barebones.com**

TextWrangler is a trademark of, and BBEdit and “It Doesn’t Suck” are registered trademarks of Bare Bones Software, Inc.

Information in this document is subject to change without notice and does not represent a commitment on the part of the copyright holder. The software described in this document is furnished under a license agreement. Warranty and license information is included in printed form with the CD-ROM package or in electronic form for downloaded products, and is presented on the next page of this user manual.

The owner or authorized user of a valid copy of TextWrangler may reproduce this publication for the purpose of learning to use such software. No part of this publication may be reproduced or transmitted for commercial purposes, such as selling copies of this publication or for providing paid for support services.

Macintosh, Mac OS, Mac OS X, Power Macintosh, and AppleScript are trademarks of Apple, Inc. Intel is a registered trademark of Intel Corporation. All other trademarks are the property of their respective owners.

TextWrangler License Agreement:

You, the Licensee, assume responsibility for the selection of the program TextWrangler to achieve your intended results, and for the installation, use, and results obtained from the program. Breaking the package seal and installing the program constitutes your acceptance of these terms and conditions. If you do not accept these terms and conditions, then do not break the package seal or install the software.

License:

You may use the program and documentation on any desired number of machines and copy the program and documentation into any machine-readable or printed form for backup or support of your use of the program and documentation on those machines, provided that no copy of the program and documentation may be used by anyone other than you.

Your use of the program and documentation is limited solely to internal use. Without limiting the generality of the foregoing, you may not, directly or indirectly, transfer, convey, distribute, or provide the program or access to the program to any third party, whether by means of a bundling, publishing, or hosting arrangement, or otherwise and whether or not for money or other consideration, without the express prior written consent of Bare Bones Software, Inc.

You may not use or copy the program or documentation, or any copy thereof, in whole or in part, except as provided in this Agreement.

You also may not modify the program or documentation, or any copy thereof, in whole or in part. If you use, copy, modify, distribute, or transfer the program or documentation, or any copy thereof, in whole or part, except as expressly provided for in this Agreement, your license is automatically terminated.

Term:

The license is effective on the date you accept this Agreement, and remains in effect until terminated as indicated above or until you terminate it. If the license is terminated for any reason, you agree to destroy the program and documentation, together with all copies thereof, in whole or in part, in any form, and to cease all use of the program and documentation.

Limited Warranty and Limitation of Remedies:

The program, documentation and any support from Bare Bones Software, Inc., are provided “as is” and without warranty, express and implied, including but not limited to any implied warranties of merchantability and fitness for a particular purpose. In no event will Bare Bones Software, Inc. be liable for any damages, including lost profits, lost savings, or other incidental or consequential damages, even if Bare Bones Software, Inc. is advised of the possibility of such damages, or for any claim by you or any third party.

General Terms:

This Agreement can only be modified by a written agreement signed by you and Bare Bones Software, Inc. and changes from the terms and conditions of this Agreement made in any other manner will be of no effect. If any portion of this Agreement shall be held invalid, illegal, or unenforceable, the validity, legality, and enforceability of the remainder of the Agreement shall not in any way be affected or impaired thereby. This Agreement shall be governed by the laws of The Commonwealth of Massachusetts, without giving effect to conflict of laws provisions thereof. As required by United States export regulations, you shall not permit export of the program or any direct products thereof to any country to which export is then controlled by the United States Bureau of Export Administration, unless you have that agency's prior written approval.

Use of the program and documentation by military and civilian offices, branches or agencies of the U.S. Government is restricted in accordance with the applicable Federal Acquisition Regulations (under which the program and documentation constitute “restricted computer software” that is “commercial computer software”) or Department of Defense Federal Acquisition Regulations Supplement (under which the program and documentation constitute “commercial computer software” and “commercial computer software documentation”) to that consistent with only those rights as are granted pursuant to the terms and conditions hereof.

Acknowledgment:

You acknowledge that you have read this agreement, understand it, and agree to be bound by its terms and conditions. You further agree that it is the complete and exclusive statement of the agreement between you and Bare Bones Software, Inc. which supersedes all proposals or prior agreements, oral or written, and all other communications between you and Bare Bones Software, Inc. relating to the subject matter of this agreement.

Info-ZIP License

This is version 2009-Jan-02 of the Info-ZIP license. The definitive version of this document should be available at ftp://ftp.info-zip.org/pub/infozip/license.html indefinitely and a copy at http://www.info-zip.org/pub/infozip/license.html.

Copyright ©1990-2009 Info-ZIP. All rights reserved.

For the purposes of this copyright and license, “Info-ZIP” is defined as the following set of individuals:

Mark Adler, John Bush, Karl Davis, Harald Denker, Jean-Michel Dubois, Jean-loup Gailly, Hunter Goatley, Ed Gordon, Ian Gorman, Chris Herborth, Dirk Haase, Greg Hartwig, Robert Heath, Jonathan Hudson, Paul Kienitz, David Kirschbaum, Johnny Lee, Onno van der Linden, Igor Mandrichenko, Steve P. Miller, Sergio Monesi, Keith Owens, George Petrov, Greg Roelofs, Kai Uwe Rommel, Steve Salisbury, Dave Smith, Steven M. Schweda, Christian Spieler, Cosmin Truta, Antoine Verheijen, Paul von Behren, Rich Wales, Mike White.

This software is provided “as is”, without warranty of any kind, express or implied. In no event shall Info-ZIP or its contributors be held liable for any direct, indirect, incidental, special or consequential damages arising out of the use of or inability to use this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the above disclaimer and the following restrictions:

- Redistributions of source code (in whole or in part) must retain the above copyright notice, definition, disclaimer, and this list of conditions.
- Redistributions in binary form (compiled executables and libraries) must reproduce the above copyright notice, definition, disclaimer, and this list of conditions in documentation and/or other materials provided with the distribution. The sole exception to this condition is redistribution of a standard UnZipSFX binary (including SFXWiz) as part of a self-extracting archive; that is permitted without inclusion of this license, as long as the normal SFX banner has not been removed from the binary or disabled.
- Altered versions--including, but not limited to, ports to new operating systems, existing ports with new graphical interfaces, versions with modified or added functionality, and dynamic, shared, or static library versions not from Info-ZIP--must be plainly marked as such and must not be misrepresented as being the original source or, if binaries, compiled from the original source. Such altered versions also must not be misrepresented as being Info-ZIP releases--including, but not limited to, labeling of the altered versions with the names “Info-ZIP” (or any variation thereof, including, but not limited to, different capitalizations), “Pocket UnZip,” “WiZ” or “MacZip” without the explicit permission of Info-ZIP. Such altered versions are further prohibited from misrepresentative use of the Zip-Bugs or Info-ZIP e-mail addresses or the Info-ZIP URL(s), such as to imply Info-ZIP will provide support for the altered versions.

Info-ZIP retains the right to use the names “Info-ZIP,” “Zip,” “UnZip,” “UnZipSFX,” “WiZ,” “Pocket UnZip,” “Pocket Zip,” and “MacZip” for its own source and binary releases.

Contents

Chapter 1 Welcome to TextWrangler 19

Getting Started 19
What Is TextWrangler? 20
How Can I Use TextWrangler? 20
Editing Source Code – 20
Editing Text Files – 20
Human Interface Notes 21
Dynamic Menus – 21
Bypassing Options Dialogs – 22
Keyboard Shortcuts for Commands – 22
Contextual Menus – 22
Dialog Box and Sheet Key Equivalents – 22
Feature Highlights 22
Info on New Features – 23
Discussion Group 23
Support Services 23
How to contact us – 24

Chapter 2 Installing TextWrangler 25

Basic Installation 25
System Requirements – 25
Installing TextWrangler – 25
Automatic Relocation – 26
First Run Configuration – 26
Checking for Updates – 27
Upgrading from a Previous Version – 27
Converting from TextWrangler to BBEdit – 27
TextWrangler's Application Support Folder 27
Application Support Folder Contents – 28
Attachment Scripts – 28
Auto-Save Recovery – 28
Color Schemes – 28
Language Modules – 28
Menu Scripts – 29
Plug-Ins – 29
Readme.txt [file] – 29
Scripts – 29
Setup – 29
Shutdown Items – 30
Startup Items – 30
Stationery – 31
Text Filters – 31
Superseded App Support Folders – 31

Preference Files and Folders 32
TextWrangler Preferences File – 32
TextWrangler Preferences Folder – 32

Chapter 3 Working with Files 33

Launching TextWrangler 34
Startup Items – 34
Creating and Saving Documents 35
Saving a Copy of a File – 36
File Saving Options – 36
File State – 37
EditorConfig – 38
Emacs Local Variables – 39
Saving with Authentication – 39
Saving Compressed Files as bz2 or gzip – 40
Crash Auto-Recovery 40
Opening Existing Documents 40
Front Window versus Separate Windows – 41
Choosing the Encoding for a Document – 41
Using the Open Command – 42
Reload from Disk – 43
Opening and Viewing Files within Zip Archives – 44
Opening bz2, gzip, and tar Files and Binary plists – 44
Opening Hidden Files – 44
Using the Open from FTP/SFTP Server Command – 44
Using the Open Selection Command – 44
Using the Open File by Name Commands – 45
Using the Open Counterpart Command – 47
Using the Open Recent Command – 47
Using the Reopen using Encoding Command – 47
Quitting TextWrangler 47
An International Text Primer 48
International Text in TextWrangler – 48
Unicode – 48
Saving Unicode Files – 49
Opening Unicode Files – 50
Accessing FTP/SFTP Servers 50
Opening Files from FTP/SFTP Servers – 50
Saving Files to FTP/SFTP Servers – 53
Using TextWrangler from the Command Line 55
Using Stationery 55
Hex Dump for Files and Documents 56
Making Backups 56
Printing 57
Printing Options – 57

Chapter 4 Editing Text with TextWrangler 59

Basic Editing 60
Moving Text – 60
Multiple Clipboards – 61
Drag and Drop – 61

Multiple Undo 62
Window Anatomy 63
The Toolbar – 63
The Split Bar – 65
The Navigation Bar – 65
The File List – 69
The Status Bar – 71
The Gutter and Folded Text Regions – 73
The View Menu 75
Text Display – 75
Show/Hide Toolbar – 76
Show/Hide Navigation Bar – 76
Show/Hide Editor – 76
Show/Hide Files – 76
Show/Hide Currently Open Documents – 76
Show/Hide Recent Documents – 76
Balance – 76
Balance & Fold – 76
Fold Selection – 76
Unfold Selection – 77
Collapse Enclosing Fold – 77
Collapse All Folds – 77
Expand All Folds – 77
Previous Document/Next Document – 77
Move to New Window – 77
Open in Additional Window – 77
Reveal in Finder – 77
Go Here in Terminal – 77
Go Here in Disk Browser – 78
Cursor Movement and Text Selection 78
Clicking and Dragging – 78
Arrow Keys – 79
CamelCase Navigation – 79
Rectangular Selections – 79
Working with Rectangular Selections – 80
Scrolling the View – 83
The Delete Key – 83
The Numeric Keypad – 83
Line Number Command – 84
Function Keys – 84
Resolving URLs – 85
Text Options 85
Editing Options – 86
Display Options – 86
How TextWrangler Wraps Text 88
Soft Wrapping – 89
Hard Wrapping – 90
The Insert Submenu 93
Inserting File Contents – 93
Inserting File & Folder Paths – 93
Inserting a Folder Listing – 93
Inserting a Page Break – 93
Inserting Time Stamps – 93
Inserting an Emacs Variable Block – 94

Comparing Text Files 94
Compare Against Disk File – 96
Multi-File Compare Options – 97
When comparing folders with the Find Differences command, TextWrangler applies any specified file filter to the contents of the resulting “Only in new” and “Only in old” lists, so that only those files that match the filter criteria will appear in the lists. 98
Using Markers 98
Setting Markers – 99
Clearing Markers – 99
Using Grep to Set Markers – 99
Spell Checking Documents 100
Check Spelling As You Type – 100
Manual Spell Checking – 101
The Spelling Panel – 101
– 102

Chapter 5 Text Transformations 103

Text Menu Commands 103
Apply Text Filter – 103
Apply Text Filter <last filter> – 104
Exchange Characters – 104
Change Case – 104
Shift Left / Shift Right – 105
Un/Comment Selection – 105
Hard Wrap – 105
Add Line Breaks – 106
Remove Line Breaks – 106
Convert to ASCII – 106
Educate Quotes – 106
Straighten Quotes – 106
Add/Remove Line Numbers – 106
Prefix/Suffix Lines – 107
Sort Lines – 107
Process Duplicate Lines – 108
Process Lines Containing – 109
Increase and Decrease Quote Level – 110
Strip Quotes – 110
Zap Gremlins – 110
Entab – 112
Detab – 112
Normalize Line Endings – 112

Chapter 6 Windows & Palettes 113

Window Menu 113
Minimize Window – 113
Bring All to Front – 113
Palettes – 113
Save Default <type of >Window – 115
Cascade Windows – 115
Arrange – 115
Cycle Through Windows – 115
Exchange with Next – 115
Synchro Scrolling – 115
Window Names – 115
Zoom (key equivalent only) – 116

Chapter 7 Searching 117

Search Windows 117
Basic Searching and Replacing 118
Search Settings – 120
Special Characters – 120
Multi-File Searching 121
Starting a Search – 122
Find All and Multi-File Search Results – 123
Specifying the Search Set – 124
Saved Search Sources – 126
Multi-File Search Options – 126
File Filters – 126
Searching SCM Directories – 128
Multi-File Replacing 129
Live Search 130

Search Menu Reference 130
Find – 130
Multi-File Search – 131
Search in [Document's Folder] – 131
Search in ... (Disk or Results Browser) – 131
Live Search – 131
Find Next/Previous – 131
Find All – 131
Find Selected Text/Previous Selected Text – 131
Use Selection for Find – 132
Use Selection for Find (grep) – 132
Use Selection for Replace – 132
Use Selection for Replace (grep) – 132
Replace – 132
Replace All – 132
Replace to End – 132
Replace & Find Again – 132
Find Differences – 132
Compare Two Front Documents – 133
Compare Against Disk File – 133
Compare Against Previous Version – 133
Apply to New – 133
Apply to Old – 133
Compare Again – 133
Find in Reference – 133
Go Menu Reference 134
Line Number – 134
Center Line – 134
Named Symbol – 134
Functions – 134
The following four commands will be available in any document which contains recognizable functions. – 134
Reveal Start/End – 134
Go to Previous/Next – 134
Markers – 135
Jump Points – 135
Previous – 135
Next – 135
Set – 135

Chapter 8 Searching with Grep 137

What Is Grep or Pattern Searching? 138

Writing Search Patterns 138
Most Characters Match Themselves – 138
Escaping Special Characters – 138
Wildcards Match Types of Characters – 140
Character Classes Match Sets or Ranges of Characters – 142
Matching Non-Printing Characters – 143
Other Special Character Classes – 144
Quantifiers Repeat Subpatterns – 145
Combining Patterns to Make Complex Patterns – 146
Creating Subpatterns – 146
Using Backreferences in Subpatterns – 147
Using Alternation – 148
The "Longest Match" Issue – 148
Non-Greedy Quantifiers – 149
Writing Replacement Patterns 150
Subpatterns Make Replacement Powerful – 150
Using the Entire Matched Pattern – 150
Using Parts of the Matched Pattern – 151
Case Transformations – 152
Examples 153
Matching Identifiers – 153
Matching White Space – 153
Matching Delimited Strings – 154
Marking Structured Text – 154
Marking a Mail Digest – 155
Rearranging Name Lists – 155
Advanced Grep Topics 155
Matching Nulls – 156
Backreferences – 156
POSIX-Style Character Classes – 157
Non-Capturing Parentheses – 158
Perl-Style Pattern Extensions – 159
Comments – 159
Pattern Modifiers – 160
Positional Assertions – 161
Conditional Subpatterns – 163
Once-Only Subpatterns – 164
Recursive Patterns – 166
Recommended Books and Resources 167

Chapter 9 Browsers 169

Browser Overview 169
List Pane – 169
Toolbar – 170
Text View Pane – 170
Splitter – 170
Disk Browsers 171
Disk Browser Controls – 171
Contextual Menu Commands – 172
Dragging Items – 172
Using the List Pane in Disk Browsers – 172
Search Results Browsers 173
Error Results Browsers 174

The Preferences Window 175
Searching the Preferences – 177
Restore Defaults – 177
Appearance Preferences 177
Toolbar – 177
Navigation Bar – 178
Editing Window – 178
Text Status Bar – 179
List Display Font – 180
Application Preferences 180
Open documents into the front window... – 180
Automatically refresh documents as they change on disk – 180
Remember the N most recently used items – 180
When TextWrangler becomes active – 181
Automatically check for updates – 181
Editing Preferences 182
Highlight instances of selected text – 182
Use "hard" lines in soft-wrapped views – 182
Soft-wrapped line indentation – 182
Line spacing – 182
Editor Defaults Preferences 183
Auto-indent – 183
Balance while typing – 183
Use typographer's quotes – 183
Auto-expand tabs – 183
Show invisible characters – 183
Check spelling as you type – 184
Default font – 184
Tab Width – 184
This option controls the default number of spaces that TextWrangler uses to represent the width of a tab character. – 184
Soft Wrap Text – 184
Keyboard Preferences 185
"Home" and "End" Keys – 185
Enter key generates Return – 185
Allow Tab key to indent text blocks – 185
Enable Shift-Delete for forward delete – 186
When auto-indenting, remove leading white space from indented line – 186
Allow Page Up and Page Down keys to move the insertion point – 186
Option-¥ on Japanese keyboards – 186
Emulate Emacs key bindings – 186
Languages Preferences 186
Installed Languages – 186
Custom Extension Mappings – 187
Menus & Shortcuts Preferences 188
Menu Key Equivalents and Item Visibility – 188
Allow menu key equivalents to autorepeat – 189

Printing Preferences 189
Print using document's font – 189
Printing font – 189
Frame printing area – 189
Print page headers – 189
Print full pathname – 189
Time stamp – 189
Print line numbers – 190
1-inch Gutter – 190
Print color syntax – 190
Text Colors Preferences 190
Selecting and Saving Color Schemes – 190
How to Change an Element's Color – 190
Language-Specific Colors – 190
Global Colors – 191
Text Encodings Preferences 191
Default text encoding for new documents – 192
If file's encoding can't be guessed, try – 192
Text Files Preferences 192
Line breaks – 192
Ensure file ends with line break – 192
Strip trailing whitespace – 193
Backups – 193
Expert preferences settings 194
The Setup Window 194
Bookmarks – 194
Filters – 194
Patterns – 195

Chapter 11 Scripting TextWrangler 197

AppleScript Overview 197
About AppleScript – 198
Scriptable Applications and Apple Events – 198
Reading an AppleScript Dictionary – 199
Recordable Applications – 204
Saving Scripts – 205
Using Scripts with Applications – 205
Scripting Resources – 206
Using AppleScripts in TextWrangler 207
Recording Actions within TextWrangler – 207
The Scripts Menu – 208
The Scripts Palette – 209
Organizing Scripts – 209
Attaching Scripts to Menu Items – 210
Attaching Scripts to Events – 211
Filtering Text with AppleScripts – 215
TextWrangler's Scripting Model 216
Script Compatibility – 216
Getting and Setting Properties – 218
Performing Actions – 219
Arranging Documents and Windows – 222
Common AppleScript Pitfalls – 224

Chapter 12 Unix Scripting and the Command-Line 225

Configuring TextWrangler 225
Syntax Coloring – 225
Switching Between Counterpart Files – 226
TextWrangler and the Unix Command Line 226
Installing the Command Line Tools – 226
The "edit" Command Line Tool – 226
The "twdiff" Command Line Tool – 227
The "twfind" Command Line Tool – 227
Unix Scripting: Perl, Python, Ruby, Shells, and more! 229
Using Unix Scripts – 229
Language Resources – 229
Setting Environment Variables for GUI Apps – 230
Line Endings, Permissions and Unix Scripts – 230
Configuring Perl – 231
Configuring Python – 231
Configuring Ruby – 231
Shebang Menu – 231
Filters and Scripts – 232
Filters – 233
Scripts – 234
Additional Notes – 234

Chapter 13 Language Modules 235

Language Modules 235
Installing Language Modules – 235
Overriding Existing Modules – 236
Codeless Language Modules – 236
Code-based Language Modules – 236
Language Module Compatibility – 236
Plug-In Obsolescence 237

Appendix A Command Reference 239

Keyboard Shortcuts for Commands 239
Assigning Keys to Menu Commands 240
Available Key Combinations – 240
Listing by Menu and Command Name 241
Listing by Default Key Equivalent 246

Appendix B Editing Shortcuts 251

Mouse Commands 251
Arrow and Delete Keys 252
Emacs Key Bindings 253
Using universal-argument – 254

Appendix C Codeless Language Modules 255

Index 257

CHAPTER

1 Welcome to TextWrangler

This chapter introduces you to TextWrangler, a high-performance text editor for the Macintosh.

In this chapter

Getting Started 19
What Is TextWrangler? 20
How Can I Use TextWrangler? 20
Editing Source Code – 20
Editing Text Files – 20
Human Interface Notes 21
Dynamic Menus – 21 • *Bypassing Options Dialogs* – 22
Keyboard Shortcuts for Commands – 22 • *Contextual Menus* – 22
Dialog Box and Sheet Key Equivalents – 22
Feature Highlights 22
Info on New Features – 23
Discussion Group 23
Support Services 23

Getting Started

Thank you for selecting TextWrangler, a high-performance text editor for the Macintosh.

If you are new to TextWrangler, we recommend that you read at least Chapters 1 through 4 of this manual to familiarize yourself with the installation and basic operation of TextWrangler. You may also wish to read or preview any other chapters that cover features you frequently use. After you have installed TextWrangler, the best way to learn it is to use it. Complete online assistance is available from the Help menu.

If you have used earlier versions of TextWrangler, we recommend that you read at least Chapter 1 for an overview of significant changes in this version, and Chapter 2 for information relevant to installation and upgrading.

What Is TextWrangler?

TextWrangler is a high-performance text editor. Unlike a word processor, which is designed for preparing printed pages, a text editor focuses on providing a means of producing and changing content. Thus, TextWrangler does not offer fancy formatting capabilities, headers and footers, graphics tools, a thesaurus, or similar staples of feature-laden "office" software. Instead, it focuses on helping you manipulate text in ways that word processors generally cannot.

In service of this goal, TextWrangler offers powerful regular expression–based ("grep") search and replace, multi-file search, sophisticated text transformations, intelligent text coloring, and other features not usually found (or missed) in word processors.

TextWrangler also has features that make it easier to edit specific kinds of text, such as source files for programming languages or HTML markup.

How Can I Use TextWrangler?

Use TextWrangler any time you need to create or edit source files, or text documents of any kind. Whether you need to find (or change!) all the occurrences of some text in a set of files, or modify or reformat large text files of any sort (or even make quick tweaks to a web page), TextWrangler is the right tool for the job.

Editing Source Code

TextWrangler is a powerful tool for editing numerous types of source code, with the following features:

- Syntax coloring helps you read your code and find simple errors.
- The function pop-up menu lets you quickly find the functions in your files.
- Find Differences lets you compare two versions of a text file and merge the differences.
- Find in Reference lets you look up documentation in the Developer Help Center.

Editing Text Files

TextWrangler is a full-featured text editor that makes it easy to create, edit, and search any sort of text file, such as release notes, articles, books, or TeX documents. It's also an excellent tool for pre- and post-processing any files that contain textual data, such as database exports or server logs.

- Grep searching lets you find and change text that matches a set of conditions that you specify.
- Multi-file search and replace lets you quickly search and modify text files anywhere on your computer.
- Numerous text manipulation and processing commands allow you to reformat or rearrange the information in text files, e.g. add or remove hard line breaks, change the case of selected text, or remove duplicate lines from a list.

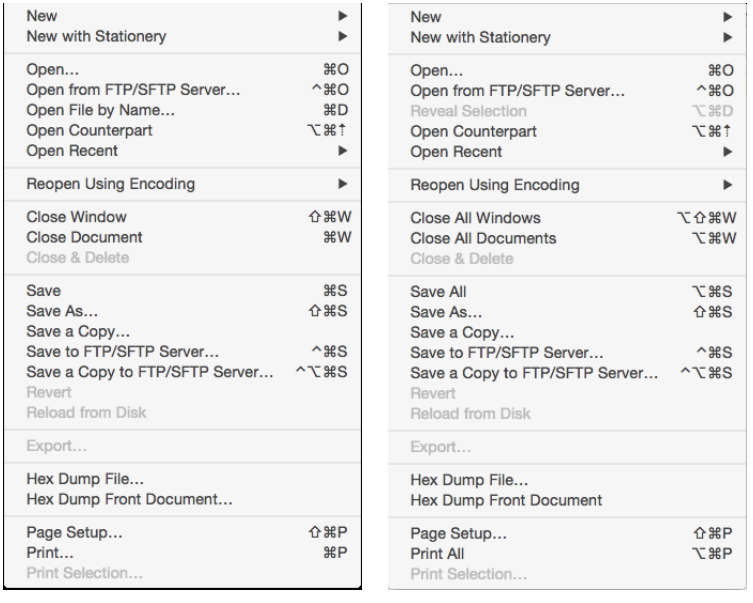
- Extensive AppleScript support allows you to combine multiple processing steps for reuse, and enables you to easily transfer data into and out of TextWrangler to other applications
- International text support lets you edit Unicode files (UTF-8 and UTF-16), as well as files saved in most non-Roman single-byte scripts.
- FTP and SFTP support lets you open and save text files located on remote servers.

Human Interface Notes

TextWrangler enhances the behavior of its menus and dialogs as described in this section.

Dynamic Menus

IMPORTANT Many of TextWrangler's pull-down menus are dynamic: if you hold down the Shift, Option, or Control key while a menu is open, you can see some of the items change. The illustration below shows what the File menu looks like normally (left) and when you hold down the Option key (right).



You can use the Shift, Option, or Control keys when you choose an item from a menu or when you use the Command-key equivalents.

Bypassing Options Dialogs

You may have noticed that commands that require additional settings to be made before they are performed appear on the menu with ellipses after their names. To bypass this step and use the command with its most recent settings, hold down the Option key while selecting the menu item. For example, “Zap Gremlins…” in the Text menu becomes “Zap Gremlins” when the Option key is pressed and, when chosen, will zap gremlins in the frontmost text document using the current settings.

Keyboard Shortcuts for Commands

Many of TextWrangler’s commands have keyboard shortcuts. TextWrangler lets you reassign the shortcuts for any menu item, clippings entry, plug-in, or script to suit your own way of working.

To change the keyboard shortcut for any menu command as well as any available scripts and text filters, go to the Menus & Shortcuts preference panel.

Contextual Menus

When you Control-click on selected text or at the insertion point in a text window, TextWrangler’s contextual menu will display a set of commands relevant to that location or text, as well as some appropriate standard commands (such as Cut/Copy/Paste) so you do not have to hunt around in the menu bar for them.

Dialog Box and Sheet Key Equivalents

You can use key equivalents to click buttons or select options in most of TextWrangler’s dialog boxes and sheets. Certain keys have the same meaning in all dialogs and sheets:

- Pressing either the Return or Enter key is the same as clicking the default button.
- Typing Command-period or pressing the Escape key is the same as clicking the Cancel button.
- You can use the Cut, Copy, Paste, Clear, and Select All commands (either from the Edit menu or with their Command-key equivalents) in any text field.

Feature Highlights

TextWrangler 5.0 offers many powerful features for editing and processing text and code, and for managing your work. Here are some highlights:

- New single-window Find Differences result browser!
- Support for the EditorConfig settings file convention
- Greatly expanded syntax coloring support (with room for future expansion)
- A selection of carefully crafted pre-installed color schemes
- Info popup offers live document statistics and file permissions adjustment

as well as all the powerful core features that TextWrangler is known for, including:

- Flexible multi-document editing windows
- Modeless Find and Multi-File Search windows provide a consistent, familiar interface to TextWrangler legendary search and replace capabilities
- In-window Live Search highlights and quickly jumps though matches
- Find Differences detects sub-line differences
- Live display of document statistics: character, word, and line count
- Transparent support for viewing (browsing) files within bz2, gzip, and Zip archives.

Info on New Features

In addition to these major features, TextWrangler 5 also contains numerous additional convenience features and interface refinements, as well as performance enhancements and bug fixes. For a detailed summary of changes and bug fixes, please refer to the current release notes, which are available in the TextWrangler Support section of our web site.

```
http://www.barebones.com/support/textwrangler/
  current_notes.html
```

Discussion Group

We maintain a public Google Group where our customers can discuss and share knowledge about using TextWrangler.

```
http://groups.google.com/group/textwrangler
```

Support Services

If you need information about using TextWrangler (or any of our other products) the Support area of our web site offers up-to-date details:

```
http://www.barebones.com/support/textwrangler/
```

You'll find a wide range of information there, including:

- Frequently Asked Questions (FAQs) — Answers for commonly encountered questions and solutions for known problems. We strongly recommend you check the TextWrangler FAQs before resorting to any other means of inquiry.
- Product Updates — The latest versions of our products are always available for download.

as well as access to language modules, developer info, and other materials.

How to contact us

If you have a registered copy of TextWrangler (or any other Bare Bones product), and you can't find the information you need on our web site, or if you encounter any problems with the software, please use the contact form on our web site or send email to:

`support@barebones.com`

Note We do not offer telephone support. Please refer to the support resources available on our web site for information and assistance, or contact us via email.

CHAPTER

2 Installing TextWrangler

This chapter tells you how to install TextWrangler on your Macintosh. It also describes the files TextWrangler creates, where it puts them, and how to install or remove optional components of TextWrangler.

In this chapter

Basic Installation **25**
System Requirements – 25 • *Installing TextWrangler* – 25
Checking for Updates – 27 • *Upgrading from a Previous Version* – 27
First Run Configuration – 26
TextWrangler's Application Support Folder **27**
Application Support Folder Contents – 28
Language Modules – 28 • *Menu Scripts* – 29 • *Scripts* – 29
Shutdown Items – 30 • *Startup Items* – 30 • *Stationery* – 31
Superseded App Support Folders – 31 • *Upgrading* – 31
Preference Files and Folders **32**
TextWrangler Preferences File – 32
TextWrangler Preferences Folder – 32

Basic Installation

TextWrangler is supplied as a single application file. Specific system requirements and installation instructions are described below, and the organization of TextWrangler's supporting files is described in subsequent sections.

System Requirements

IMPORTANT TextWrangler 5.0 requires Mac OS X 10.8.5 or later, and is fully compatible with OS X 10.10 "Yosemite" and OS X 10.11 "El Capitan". The software will not run on Mac OS 9 or any earlier versions of Mac OS X, and will not run on PowerPC-based machines.

Installing TextWrangler

If you download TextWrangler directly, you will receive a standard disk image (".dmg") file. Your web browser may automatically mount the disk image once the download is complete; otherwise, you should double-click on the disk image file to mount it. Once the disk image is mounted, drag the "TextWrangler" application over the adjacent icon for the Applications folder and drop it there to copy TextWrangler onto your Mac's hard drive. You can then dismount (eject) the disk image and discard the ".dmg" file.

If you obtain TextWrangler through the Mac App Store, you must install it via the App Store application.

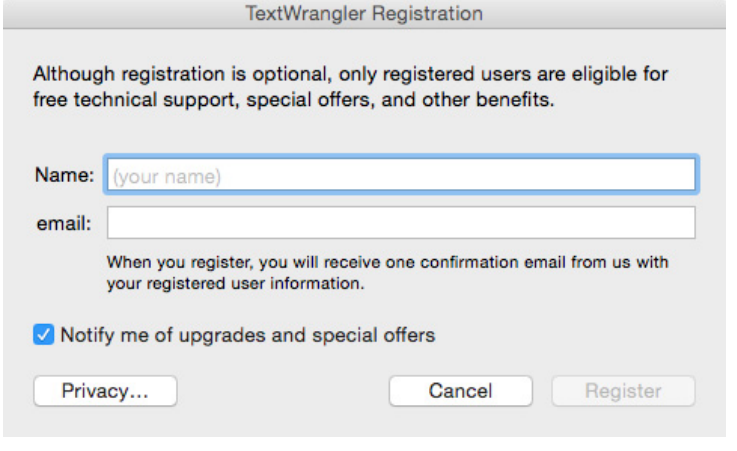
Automatic Relocation

If you launch TextWrangler from any location other than your Mac's main "Applications" folder (/Applications/) or the account-specific equivalent (~/Applications/), TextWrangler will offer to automatically relocate itself into the main "Applications" folder.

Note This behavior does not apply to copies of TextWrangler obtained from the Mac App Store.

First Run Configuration

The first time you launch TextWrangler, it will display the "Welcome to TextWrangler" dialog. This dialog allows you to choose whether to register TextWrangler, and if you choose to register, whether you want to be notified about upgrades and special offers.


TextWrangler Registration
Although registration is optional, only registered users are eligible for free technical support, special offers, and other benefits.
Name:
(your name)
email:
When you register, you will receive one confirmation email from us with your registered user information.
Notify me of upgrades and special offers
Privacy...
Cancel
Register

Note Registration is completely optional and you are under no obligation to either register or receive notices.

Checking for Updates

TextWrangler offers the option to automatically check for updates; this behavior is controlled by the "Automatically check for updates" option in the Application preferences panel. You can also directly check for updates at any time by choosing Check for Updates in the TextWrangler (application) menu.

In order to update TextWrangler when future versions become available, you need only apply the update when prompted. (Alternatively, you may quit TextWrangler, and manually replace your existing copy with the updated version.) The first time you launch a newer version of the software, TextWrangler will prompt you for any further actions which may be needed, such as updating the command-line tools.

Note Copies of TextWrangler obtained through the Mac App Store do not include the update check feature; instead, updates will be delivered through the App Store app.

Upgrading from a Previous Version

IMPORTANT If you are upgrading from any version prior to TextWrangler 3.0, in addition to installing the current application, you will need to manually copy over any items you wish to use from your existing "TextWrangler Support" folder into TextWrangler's application support folder. You should **not** simply rename your existing "TextWrangler Support" folder.

Please carefully read the remainder of this chapter, since the organization of TextWrangler's supporting files has changed considerably. We have provided specific suggestions and tips for transferring your customized support items in each category.

Converting from TextWrangler to BBEdit

If you are using TextWrangler 4.5 or later, and then install and launch BBEdit 11 for the first time (i.e. with no existing preferences), BBEdit will look for any existing TextWrangler preferences and import those instead of using its own factory defaults.

In addition, if a TextWrangler preference data folder exists (at "/Users/<username>/Library/TextWrangler/" which is the default for TextWrangler 4.5 and later), then BBEdit will copy that folder to "/Users/<username>/Library/BBEdit/".

Finally if a TextWrangler application support folder exists (at "/Users/<username>/Library/Application Support/TextWrangler/" BBEdit will copy this folder to "/Users/<username>/Library/Application Support/BBEdit/".

TextWrangler's Application Support Folder

TextWrangler's application support folder contains items which define or extend TextWrangler's capabilities, such as clippings sets, language modules, scripts, and more. These items are organized into subfolders according to their purpose (described below).

IMPORTANT TextWrangler's application support folder must be present in the following location:

```
/Users/<username>/Library/Application Support/TextWrangler/
```

If this folder does not exist when TextWrangler starts up, TextWrangler will create this folder together with a number of standard subfolders, to which you can add any appropriate items. None of these folders are essential for doing basic tasks with TextWrangler, and you can remove any or all of them that you don't use.

Note Although OS X hides your account's local "Library" folder by default, you can still access this folder in the Finder by holding down the Option key, then selecting **Library** from the **Go** menu in the menu bar.

Application Support Folder Contents

TextWrangler's application support folder contain various subfolders, each of which holds a specific type of support item.

You can relocate TextWrangler's application support folder by placing a symlink (though not an alias) in the default location which points to this folder's actual location. However, you cannot independently relocate any of the subfolders within the application support folder. We also recommend that you do not try to share scripts between TextWrangler and other applications, nor should you attempt to store TextWrangler's application support folder on any remote (server) volume.

Attachment Scripts

This folder does not exist by default, but you may create it at any time. The Attachment Scripts folder contains AppleScripts which are run at specific points: when TextWrangler starts or quits; and when documents are open, saved, and closed.

Auto-Save Recovery

[This folder's default location has moved into TextWrangler's local preference data folder.]

Color Schemes

TextWrangler will automatically create this folder when needed. The Color Schemes folder stores any custom color schemes which you have saved within the Text Colors preference panel (or which you have download and copied over). Each scheme is stored within a separate ".bbColorScheme" file.

Language Modules

TextWrangler does not create this folder by default, but will do so if necessary. The Language Modules folder allows you to add syntax coloring and function navigation support for additional languages by installing language modules.

IMPORTANT **Please do not attempt to extract or modify the language modules contained in the TextWrangler application bundle.**

A list of additional modules from third-party developers is available on our web site, or you may create your own compiled or codeless language modules (see "Codeless Language Modules" on page 236).

Upgrading You should move or copy over any compatible third-party language modules that you wish to preserve.

Menu Scripts

This folder does not exist by default, but you may create it. The Menu Scripts folder contains AppleScripts that are attached to TextWrangler menu items. (For more details on using menu scripts, please see "Attaching Scripts to Menu Items" on page 210.)

Upgrading You should move or copy over any menu scripts that you wish to preserve.

Plug-Ins

IMPORTANT TextWrangler 5 does not support plug-ins written for much older versions, and will not load any items present in this folder.

If you used any third-party commercial plug-in, please contact its developer for information on alternative solutions.

Readme.txt [file]

This file contains an abbreviated description of the default contents of TextWrangler's application support folder.

Scripts

TextWrangler will automatically create this folder if it does not exist. The Scripts folder may contain AppleScript files and executable Unix files (scripts). Items placed in this folder will appear in the Scripts menu (left), and you may place items within subfolders (up to four levels deep) to organize them.

You may run these items from the Scripts menu, the floating Scripts palette, or via assigned key equivalents. (You may use the Menus & Shortcuts preference panel to assign a key equivalent to any item in the Scripts menu.)

TextWrangler runs such items by simply loading the item and calling it directly, without providing any inputs. (Naturally, AppleScript scripts may query TextWrangler for more information, and Unix scripts may obtain information from the environment variables that TextWrangler sets, while text factories will use their stored target list if any.)

Upgrading If you are upgrading from a version prior to TextWrangler 4.5, TextWrangler 5 will copy all your existing Unix scripts into this folder the first time you launch it.

If you are upgrading from a version prior to 3.0, you must instead manually move or copy over any customized scripts that you wish to preserve. Note also that scripts written for use with such older versions of TextWrangler may no longer work. (Please see Chapters 12 and 13 for more details and tips on modifying your existing AppleScripts and Unix filters & scripts.)

Setup

TextWrangler will automatically create this folder if it does not exist. The Setup folder contains configuration data which previous versions stored in either the preferences file proper or in the "com.barebones.textwrangler.preferenceData" preference folder, including: stored file filters, FTP/SFTP bookmarks, grep patterns, and key bindings.

The Setup folder may contain any or all of the following data files.

File Filters.filefilters

TextWrangler stores all user-defined file filter patterns in this file. You should not attempt to directly edit the contents of this file; instead, please use the Filters panel of the Settings window to add, modify, or remove stored grep patterns.

FTP Bookmarks.xml

TextWrangler stores user-defined FTP and SFTP bookmarks in this file. You should not attempt to directly edit the contents of this file; instead, please use the Bookmarks panel of the Settings window to add, modify, or remove bookmarks.

Grep Patterns.xml

TextWrangler stores user-defined search patterns in this XML file. You should not attempt to directly edit the contents of this file; instead, please use the Patterns panel of the Settings window to add, modify, or remove stored grep patterns.

Upgrading If you created any custom grep patterns in a previous version, TextWrangler will import those patterns; otherwise, TextWrangler will create a default set of patterns.

Menu Shortcuts.xml

TextWrangler stores keyboard shortcuts for menu commands in this XML file.

Not Menu Shortcuts.xml

TextWrangler stores other keyboard shortcuts in this XML file.

Shutdown Items

This folder does not exist by default, but you may create it at any time. The items in this folder are opened when you quit TextWrangler. Usually, this function is used to run scripts of some sort.

Shutdown items are run after all windows have been closed, and only if TextWrangler is actually quitting. Thus, if you wish to run any items as the immediate result of a Quit command, you should write a menu script attached to TextWrangler•Quit.

Note In some previous versions of TextWrangler, shutdown items were run before all windows were closed, and were run whenever the application was told to quit (either by the Quit menu command or via the scripting interface), regardless of whether it actually quit or not.

Upgrading You should move or copy over any shutdown items that you wish to preserve.

Startup Items

This folder does not exist by default, but you may create it at any time. When launched, TextWrangler will open any items it finds in this folder.

If the items present are text files or other documents of a type that TextWrangler knows how to handle, TextWrangler will open them directly. If you place a compiled AppleScript in this folder, TextWrangler will execute the script. If you place a folder alias here, TextWrangler will open a disk browser window based at that folder.

If you place other types of items in this folder, TextWrangler will ask the Finder to open them.

Upgrading You should move or copy any file or application aliases that you wish to preserve. If you have any AppleScripts startup items, please see the preceding upgrade note for the Scripts folder about script compatibility.

Stationery

This folder does not exist by default, but you may create it at any time. The Stationery folder contains stationery files for use with TextWrangler's New with Stationery command, and the Stationery List palette. Stationery files may be placed within subfolders (up to four levels deep) to organize them.

You can hide, or show, all items included from the global folder by using the menu item "Hide/Show Library Stationery".

Upgrading You should move or copy over any stationery documents that you wish to preserve.

Text Filters

This folder does not exist by default, but you may create it at any time. The Text Filters folder can contain compiled AppleScripts and/or Unix filters, which you may apply to the frontmost document via the Apply Text Filter command in the Text menu.

When you apply such an item, TextWrangler will pass either the selected text (if any) or the contents of the entire document on STDIN to Unix executables and filters, and as a reference to a 'RunFromTextWrangler' entry point in AppleScripts. (If an AppleScript script does not have a 'RunFromTextWrangler' entry point, TextWrangler will call its run handler, again passing a reference to the current selection range.)

AppleScript scripts should return a string which TextWrangler will use to replace the selection range, while Unix filters should write to STDOUT.

Upgrading If you are upgrading from TextWrangler 4.x or older, TextWrangler 5 will copy all of your existing Unix filters into this folder the first time you launch it.

Superseded App Support Folders

Upgrading TextWrangler no longer uses the Text Factories or Unix Support folders, though these folders may exist if they were created by a version prior to 4.0. Instead, the first time you launch TextWrangler 5, it will copy all existing Unix scripts into the Scripts folder, and all existing Unix filters into the Text Filters folder.

Preference Files and Folders

When you start up TextWrangler, it may create the files and folders noted in this section.

TextWrangler Preferences File

All of TextWrangler's basic preference settings are stored in the file "/Users/<username>/Library/Preferences/com.barebones.textwrangler.plist", which is created and maintained using standard system services.

In addition to the settings documented in Chapter 10, you may adjust additional expert preference settings outside of TextWrangler by issuing suitable "defaults write" commands. For a complete list of available expert preference settings, please see the "Expert Preferences" page of the built-in Help book. (To open the Help book, choose "TextWrangler Help" in the "Help" menu.)

Upgrading TextWrangler 5 will import and use any relevant preference settings from TextWrangler 3.0 or later, provided that prefs file is present.

TextWrangler Preferences Folder

By default, TextWrangler 5 stores its ancillary preference data in the folder "/Users/<username>/Library/TextWrangler" in order to comply with OS guidelines, though if the folder "~/Library/Preferences/com.barebones.textwrangler.PreferenceData/" (used by older versions) already exists, TextWrangler 5 will continue to use it.

The standard contents of this folder are as follows.

Document State.plist

TextWrangler stores state information for individual documents in this file.

Recent Files & Favorites

This folder is no longer used and may be deleted.

Recent Folders & Favorites

This folder is no longer used and may be deleted.

Save Application State.appstate

TextWrangler stores application state info in this file.

Saved Sources.xml

TextWrangler stores all user-defined search sources in this file.

CHAPTER

3 Working with Files

This chapter discusses how to use TextWrangler to manipulate text files.

In this chapter

Launching TextWrangler **34**
Startup Items – 34
Creating and Saving Documents **35**
Saving a Copy of a File – 36 • *File Saving Options* – 36
File State – 37 • *EditorConfig* – 38
Saving with Authentication – 39
Saving Compressed Files as bz2 or gzip – 40
Crash Auto-Recovery **40**
Opening Existing Documents **40**
Choosing the Encoding for a Document – 41
Using the Open Command – 42 • *Reload from Disk* – 43
Opening and Viewing Files within Zip Archives – 44
Opening bz2, gzip, and tar Files and Binary plists – 44
Opening Hidden Files – 44
Using the Open Recent Command – 47
Using the Reopen using Encoding Command – 47
Using the Open Selection Command – 44
Quitting TextWrangler **47**
An International Text Primer **48**
International Text in TextWrangler – 48 • *Unicode* – 48
Saving Unicode Files – 49 • *Opening Unicode Files* – 50
Accessing FTP/SFTP Servers **50**
Opening Files from FTP/SFTP Servers – 50
Saving Files to FTP/SFTP Servers – 53
Using TextWrangler from the Command Line – 55
Using TextWrangler from the Command Line **55**
Using Stationery **55**
Hex Dump for Files and Documents **56**
Making Backups **56**
Printing **57**
Printing Options – 57

Launching TextWrangler

To launch TextWrangler, double-click the TextWrangler application icon or a TextWrangler document. Holding down the following keys at launch has the indicated effects, overriding any startup options set in the Application preference panel. When one of these keys is held down, TextWrangler will beep after it finishes launching.

Modifier	Function
Option	Suppress startup items only
Shift	Disable all external services and startup items, and skip reopening all documents **except** those which contain unsaved changes.
Command-Control-Shift	Disable all external services and startup items, and optionally discard auto-recover information (which will result in the loss of any unsaved changes).

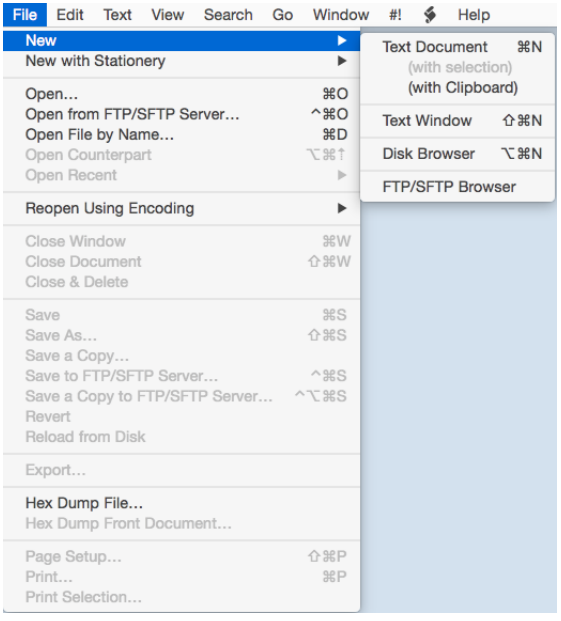
Startup Items

When launched, TextWrangler will look for a folder named Startup Items in its application support folder (see “TextWrangler’s Application Support Folder” on page 27). If this folder is found, TextWrangler will open any items it finds in the folder.

If the items present are text files or other files of a type that TextWrangler knows how to handle, TextWrangler will open them directly. If you place a compiled AppleScript in this folder, TextWrangler will execute the script. If you place a folder alias here, TextWrangler will open a disk browser window based at that folder. If you place other types of items in this folder, TextWrangler will ask the Finder to open them.

Creating and Saving Documents

To create a new text document or special-purpose window within TextWrangler, pull down the File menu and open the New submenu. Since TextWrangler uses different kinds of documents for specific purposes, you will see several options, as follows:



The available commands and their effects are as follows:

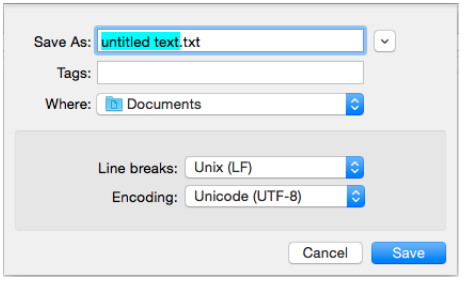
- Text Document: Opens an empty text document.
- (with selection): Opens a new text document containing any text selected in the active document and having the same display font, saving you the trouble of copying and pasting it.
- (with Clipboard): Opens a new text document and automatically pastes the contents of the current clipboard into it.
- Text Window: Opens a new text window (see “Text Windows” later in this chapter for more information).
- Disk Browser: Opens a new disk browser (see Chapter 9 for more information).
- FTP/SFTP Browser: Opens a new FTP/SFTP browser (see later in this chapter for more information).

You can also create a new text document by selecting text in any application which supports the system Services menu, and choosing the New Window with Selection command in the Text section of the Services submenu. TextWrangler will open a new text window containing a copy of the selected text.

When you want to save a new text document:

1 **Choose the Save or Save As command from the File menu.**

TextWrangler opens a standard Save sheet:



2 **Give the file a name.**

3 **Change the automatically-provided filename extension (if necessary).**

TextWrangler will automatically provide a filename extension based on the current document's language type.

4 **Change any desired options (see below).**

5 **Click Save to save the file.**

You can also create a new document from the selected text in any open window with TextWrangler's contextual menu. Simply Control-click the selected text and choose New (with selection) or Save Selection from the menu that appears. Depending on which command you choose TextWrangler will either create a new editing window containing the selected text, or display the Save dialog and allow you to create a new file containing the selected text. The new file will use the same options (see "File Saving Options," below) as those of the original parent document.

Saving a Copy of a File

You can save a copy of a file with TextWrangler's Save a Copy command in the File menu. Just like the Save As command, the Save a Copy command displays a Save dialog and lets you choose a name and location for the file. However, unlike the Save As command, where TextWrangler will start working with the new file you saved in place of the original, when you use Save a Copy, you create a new file in the designated location, but keep working with the original file.

For example, say you are editing a document called Test.py and use the Save a Copy command to save a document called Backup-Test.py. The next time you choose the Save command, TextWrangler saves the changes to Test.py and not to Backup-Test.py.

File Saving Options

TextWrangler's Save sheet is the standard Macintosh Save sheet with these additions:

Line Breaks

The Line Breaks menu let you choose what kinds of line breaks TextWrangler writes when you save the file. Choose:

- Unix line breaks (ASCII 10) for most purposes, including use with modern Mac applications, or for files being saved to a Unix file server. This is the default option.
- Classic Mac line breaks (ASCII 13) if you will be using the file with other applications which expect this format.
- Windows line breaks (ASCII 13/10) if the file resides on a Windows file server or if you will be sending it to someone who uses a Windows- or DOS-based system.

Encoding

TextWrangler lets you save documents using any character set encoding supported by Mac OS X, including a variety of Unicode formats (see "Saving Unicode Files" on page 49). To select an encoding, choose its name from the Encoding pop-up menu. The list of available encodings is controlled by your preference settings (see "Text Encodings Preferences" on page 191).

When you select an encoding that requires a Unicode file format, you can also choose "Unicode" as an option from the Line Breaks pop-up menu in this dialog. (Unicode has its own line-ending standard.)

UTF-16 files created by TextWrangler are given a type of 'utxt'—the Mac standard type for Unicode text files. UTF-8 files are given a type of 'TEXT' for compatibility with other applications; however, TextWrangler will also recognize such files with type 'UTF8'.

Note You can choose which encodings appear in the Encoding pop-up menu in the Text Encodings preference panel.

File State

If you modify a document's window position or display settings and then save the document, TextWrangler stores state information, which it will use to reopen that document in the same manner.

TextWrangler captures only those settings which are fundamental to the document (window position, selection range, folds, splitter setting), or any settings which vary from the global preferences. (The latter ensures that changes to the global preferences are never inappropriately overridden by stored display options derived from prior global or default preference settings.)

For example, say TextWrangler is currently configured to use Courier as its default display font, and you open (or create), save, and close a document which uses that font. If you then change TextWrangler's default display font to Menlo before reopening that document, the document will display in Menlo.

Note The above example uses the display font option for illustration, but the same principle applies to any document display option which derives from TextWrangler's global preferences.

EditorConfig

TextWrangler now supports the 'EditorConfig' settings file convention. You can learn more about this convention at the EditorConfig project website:

`http://www.editorconfig.org/`

TextWrangler supports most, but not all, of the core EditorConfig properties listed here:

`https://github.com/editorconfig/editorconfig/wiki/EditorConfig-Properties`

except that the 'end_of_line' and 'max_line_length' properties are **not** supported.

In addition to the core EditorConfig properties, BBEdit and TextWrangler jointly support a number of application-specific additions. First, they support the following keys which originated as Emacs variables:

- coding: similar to 'charset', but allows you to specify any IANA character set name.
- mode: allows you to explicitly specify the language. Many of the Emacs-style mode names work, as long as they correspond to supported languages. In addition, any installed language may be expressed as a mode name by lowercasing its name and replacing spaces with dashes. For example, "Ruby in HTML" becomes 'ruby-in-html'; or "Strings File" becomes 'strings-file'.
- make-backup-files: set to 1 or 0, determines whether TextWrangler makes a backup of the file when saving.
- backup-inhibited: if present and set to 1, will explicitly suppress the creation of backup files when saving.

Finally, BBEdit and TextWrangler support some custom keys which map specifically to their own internal settings. All of these keys have names that begin with 'x-' in order to prevent collisions with any future core keys. These correspond directly to individual document settings, and if present will override TextWrangler's global preferences:

- x-typographers-quotes: Use Typographer's Quotes
- x-balance-while-typing: Balance While Typing
- x-soft-wrap-text: Soft Wrap Text
- x-soft-wrap-mode: (string) Must be one of 'CharacterWidth', 'WindowWidth', or 'PageGuide'
- x-soft-wrap-limit: (integer) if the wrap mode is 'CharacterWidth', specifies the number of characters
- x-font-name: (string) the display font name
- x-font-size: (integer) the display font point size
- x-show-invisibles: Show Invisibles
- x-show-spaces: Show Spaces

Unless otherwise noted, these application-specific keys are all Boolean flags.

Emacs Local Variables

Emacs (the popular Unix text editor) supports a convention in which you can define Emacs-specific settings in a block of text near the end of the file, or in the first line of the file. This convention helps maintain consistency when sharing files among a group of people, or across multiple systems.

For general information on Emacs variables, please see the GNU Emacs manual:

```
http://www.gnu.org/software/emacs/manual/html_node/emacs/
Specifying-File-Variables.html
```

TextWrangler will read and honor the “coding”, “tab-width”, and “x-counterpart” variables in any file which contains an Emacs variable block, and adjusts the value of the “coding” variable if you change the document’s encoding by using the Encoding popup.

If a file contains an Emacs variable block (or line) having a “mode” variable, TextWrangler will attempt to match the mode name against all currently recognized languages, before attempting to match the file name suffix or guess based on the file's contents.

You may add an Emacs variable block (or lines) to any document either directly, or by selecting the Emacs Variable Block command from the Insert submenu of the Edit menu.

Here is an example variable block from a plain text file:

```
Local Variables:
coding: ISO-8859-1
tab-width: 8
End:
```

You may also add the TextWrangler-specific variable “make-backup-files” to control whether or not TextWrangler should back up a given file. For more details, please see “Controlling Backups with Emacs Variables” on page 193.

Saving with Authentication

TextWrangler supports saving files that require administrator privileges, if you possess the necessary user and password information to enable this. For example, you can edit and save files that are owned by, and only readable by, the “root” user. Authenticated saving is particularly useful in conjunction with the “Show Hidden Items” option in the Open dialog, which allows you to see and open files in hidden folders (like /bin and /usr).

When you open a file for which you do not have write privileges, TextWrangler will display a slash through the pencil icon in the toolbar. To edit the file, click the pencil icon. TextWrangler will prompt you to confirm whether you wish to unlock the file. (Option-click the pencil icon to skip the confirmation dialog.)

When you are finished editing, simply choose Save from the File menu. TextWrangler will prompt you to authenticate as a user with administrator privileges. Type a suitable user name and password to save the file.

Saving Compressed Files as bz2 or gzip

TextWrangler transparently supports opening, browsing, and saving files compressed in the 'bz2' and 'gzip' formats. To save a file with gzip compression, simply append an filename extension of ".bz2", ".gz", or ".gzip" when creating it (or doing a Save As of an existing document). (For more information on these formats, issue the commands 'man bz2' or 'man gzip' in the Terminal.)

Crash Auto-Recovery

IMPORTANT TextWrangler automatically saves auto-recovery information for all unsaved open documents at the specified interval. When you relaunch TextWrangler after a system or application crash, TextWrangler will reopen and restore the contents of any documents for which recovery information is available.

TextWrangler's auto-recovery mechanism can help minimize the chance of data loss in the event of unexpected system or application crashes. However, it may not protect against extraordinary events, and it will not protect against hardware failures or any other events that render your disk unreadable. You should always manually save a document after making any significant changes to it, and we strongly recommend that you take appropriate measures to back up your important files and other data.

Opening Existing Documents

There are several ways to open existing documents with TextWrangler.

- Double-click any file with a TextWrangler document icon.
- If TextWrangler is running, choose the Open or Open Recent command from the File menu.
- Select the name of a file in a TextWrangler editing window; then use the Open Selection command in the File menu.
- Double-click a file name in a browser's file list (see Chapter 9, "Browsers").
- Drag a file's icon to the Windows palette (see Chapter 6, "Working with Windows").
- Drag a file's icon to the file list of any editing window (see Chapter 4, "Window Anatomy).
- Drag a file's icon to the TextWrangler icon or to an alias of the icon.
- Select a file in the Finder, and use the Open File in TextWrangler command in the File section of the Services submenu.

TextWrangler can natively open files with type 'TEXT', 'utxt', and 'UTF8'. By default, TextWrangler will attempt to display the contents of image files via QuickTime, but will open PDF files in a "raw" condition as if they were text documents. You can adjust how TextWrangler should handle such files via its expert preferences. (See the "Expert Preferences" page of TextWrangler's built-in Help book for complete details.)

Front Window versus Separate Windows

Since TextWrangler supports opening multiple documents into a single text window, you must decide whether the application should work in this manner, or whether it should instead open each document into its own window. (A document may represent either a file which you open for editing, a text document created by the New Document command on the File menu, or any similar item, such as a text document created via the scripting interface.)

By default, TextWrangler will open all new documents into a single text window, but you can instead configure it to open each document into its own text window by turning off the option "Open documents into the front window when possible" in the Application preference panel.

Choosing the Encoding for a Document

When you open a document, TextWrangler will automatically examine its contents for any indication of the proper encoding, and attempt to handle it appropriately. If TextWrangler cannot determine the proper encoding, and you opened the file with the Open command, it uses the encoding specified in the Read As pop-up menu on the Open dialog. Otherwise, it uses the encoding specified by the "If the file's encoding can't be guessed, use" preference setting in the Text Encodings preference panel.

Note You can choose which encodings appear in the Read As pop-up menu by using the Text Encodings preference panel.

Here are the details of the steps that TextWrangler goes through to determine the proper encoding for a file:

1. **If the file is well-formed HTML or XML, TextWrangler looks for an "encoding=" or <meta charset=> directive.**
2. **If the file contains an Emacs variable specifying its encoding, TextWrangler will use that encoding.**
3. **If you have opened the file with TextWrangler before, TextWrangler will use the file's stored encoding info (if any).**
4. **If the file contains a UTF-8 or UTF-16 (Unicode) byte-order mark (BOM), TextWrangler opens it as that type of Unicode file.**
5. **If the file has a resource that contains font information (such as a 'styl' resource) and that resource specifies a multi-byte font, TextWrangler opens the file as a Unicode file.**
6. **If you are opening the file with the Open command, TextWrangler uses the encoding specified Read As pop-up menu on the Open dialog.**
7. **If the file contains no other cues to indicate its text encoding, and its contents appear to be valid UTF-8, TextWrangler will open it as UTF-8 without recourse to the below preferences option.**
8. **Finally, it uses the encoding chosen for the option "If the file's encoding can't be guessed, use" from the pop-up menu in the Text Encoding preference panel.**

To change the encoding for a file after opening it, use the Text Encoding popup in the document's status bar.

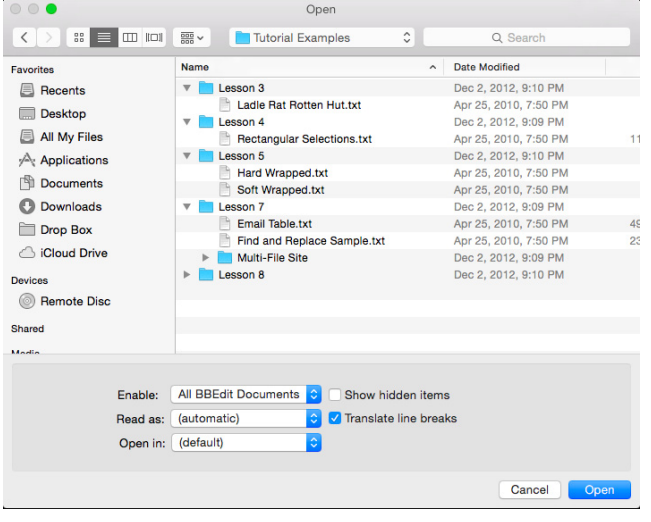
Note If an encoding change results in the conversion of a document's contents from a single-byte script to a multi-byte script, TextWrangler will mark the document as being "dirty" or changed.

Using the Open Command

To open a file with the Open command:

1 **Choose Open from the File menu.**

TextWrangler displays the Open dialog box:



2 **Select the file you want to open.**

You can select (or deselect) multiple files by holding down the Command key or the Shift key as you click the files.

3 **Change any desired options (see below).**

4 **Click Open to open the file.**

You can use the options described below when you open a file.

Enable Menu

This popup menu lets you choose what kinds of files can be selected in the Open dialog. If you know a file contains text, but it does not appear in the Open dialog, this means the file does not have a recognized filename extension or any other property which would allow TextWrangler or the system to recognize that it contains text. This is sometimes the case with files received from other computers or downloaded from the Internet. Choose "Everything" to view all available files without restriction.

TextWrangler will first check the file's suffix against its own language mappings (as shown in the Languages preference panel) and if the suffix matches up with any language (even if that language is "None"), TextWrangler will assume that file to be a text file. Thus, you can use TextWrangler's suffix mappings to make it to treat any file as text which the system does not recognize. If TextWrangler cannot match a file's suffix (or its name) to a known language, TextWrangler will next check to see if the system recognizes that file's content type.

Show Hidden Items

When this option is selected, the Open dialog will display invisible files and folders. The setting you choose will persist until you change it.

Translate Line Breaks

When this option is selected, TextWrangler translates Windows or Unix line breaks when opening a file. Otherwise, TextWrangler leaves the original line breaks untranslated.

Unlike the other options in the Open dialog, the setting of this option is not preserved between uses of the Open command, since in general you will only need to use this operation temporarily, e.g. to read in a particular file.

Read As

When opening a file, you can tell TextWrangler what encoding to use by choosing it from this pop-up menu. Usually, TextWrangler will correctly auto-detect the encoding, but if it does not, you can try applying the Reopen Encoding command with an appropriate encoding. Chapter 5 includes more information on encodings.

Open In

When opening one or more files, you can use the options on this pop-up menu to override your default document opening preferences. These options have the following effect:

- (default): TextWrangler will open the selected documents according to your preference settings.
- Front Window: TextWrangler will open all of the selected documents into the frontmost text window. If there are no text windows open, or the frontmost text window contains an active sheet, this option will be disabled.
- New Window: TextWrangler will open all of the selected documents into a new text window.
- Separate Windows: TextWrangler will open each of the selected documents into its own text window.

Reload from Disk

When you choose this command, TextWrangler will compare the contents of the current document in memory to those of its file on disk, and reload the document from its file if they differ. This is useful in situations where the file may have changed without TextWrangler noticing, which can happen if, e.g. the "Automatically refresh documents" option in the Application prefs panel is turned off, or if the file is on a shared disk and has been modified from another workstation.

Opening and Viewing Files within Zip Archives

TextWrangler can transparently open and display the contents of most Zip-compressed archives (".zip") either directly or during a multi-file search. (Zip archives must be in the format created by the Finder's "Compress" command, or by applying `ditto -k` from the command line.)

Note If the Zip archive contains only one top-level item, and that item is a folder, TextWrangler will "hoist" the rest of that package's contents and not display the top-level item.

Opening bz2, gzip, and tar Files and Binary plists

TextWrangler transparently opens and displays the contents of any bz2 or gzip-compressed files (".bz2", ".gz" and ".gzip" files), as well as tarballs (".tar" files) and binary plists (".plist" files), both directly and during multi-file search.

This is especially useful for viewing and working with system log files and similar automatically-generated files, as well as system and application preference files.

If you make any changes to such a file and save it, TextWrangler will automatically re-compress or re-convert the file on save.

Opening Hidden Files

Turn on the "Show Hidden Items" option in the Open dialog to display hidden files (including both files whose invisible attribute has been set, and those whose names begin with a period) or files from a folder which is normally hidden by the system.

Using the Open from FTP/SFTP Server Command

See "Accessing FTP/SFTP Servers" on page 50.

Using the Open Selection Command

The Open Selection command lets you quickly invoke the Open File by Name command to search for any file that is referenced in the text of a document. It is particularly useful for opening include files or any document referenced by another file.

To open a file whose name is referenced in the text of a document:

1. **Select the file name within the body of the document.**
2. **Choose Open Selection from the File menu.**

   If a suffix of the form ".x" follows the name, TextWrangler will automatically expand the selection to include the suffix.

   TextWrangler will display the Open File by Name window, prepopulated with the selected text.
3. **Click Open or type Return in the Open File by Name window.**

TextWrangler also understands the Unix-style line number and character offset specifications ":line:offset" that can be appended to a file name, and will honor them when opening a file. If the specified file is already open, this command will simply select the designated location within the file. (These specifications are frequently generated by Unix command line tools.)

For example, selecting the text "main.cp:210" and choosing Open Selection will bring up the Open File by Name window prefilled with that search string, and when you click Open, TextWrangler will then open the file "main.cp" and automatically select line 210. Likewise if you apply Open Selection to the text "foo.cp:398:43" and invoke Open File by Name, TextWrangler will open the file "foo.cp" and automatically position the insertion point at the specified location.

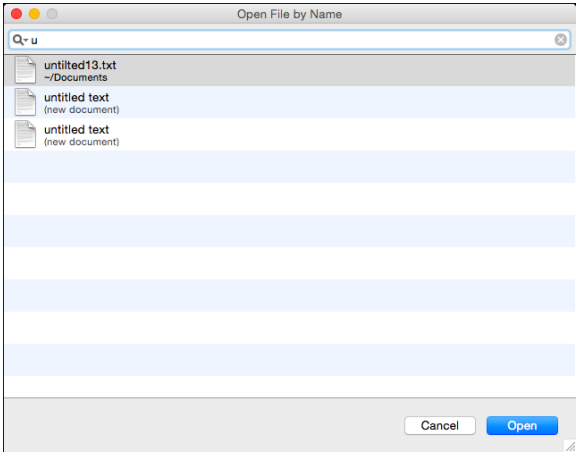
In searching for the requested file, TextWrangler will look in the following locations (in order of preference):

- If there is a disk browser open, TextWrangler will search within its current root directory.
- Otherwise, TextWrangler will look first in the same folder as the file containing the selected file name, and then in any subfolders within that folder.
- If TextWrangler cannot find the file in any of these places, it will display a Choose Folder dialog to allow you to locate the file manually.

In some cases, there may be more than one file with the same name in the various folders TextWrangler looks in. Normally, TextWrangler opens the first such file it encounters, and then stops.

Using the Open File by Name Commands

If there is no selection, or there is no text display view in the front window, Open Selection becomes Open File by Name. Choosing this command brings up the Open File by Name window.)



Activating the Open File by Name window, or choosing the menu command, will place keyboard focus in the search box and select its contents, so that you can just start typing. (To clear an existing entry, click the “clear” widget at the right-hand edge of the field.)

As you type, TextWrangler will search for files matching the current string as well as wildcard matches, and present a list of possible matches in the bottom panel of the window. If the string you enter contains wildcard characters (see below) then TextWrangler will treat it as a wildcard pattern. If the string you enter does **not** contain wildcards, TextWrangler will instead use it as a basis for casting a wide net.

TextWrangler will look for matches in the following locations (in order of preference):

- If there is a disk browser open, TextWrangler will search within its current root directory.
- Otherwise, TextWrangler will look first in the same folder as the file containing the selected file name, and then in any subfolders within that folder.
- If TextWrangler cannot find the file in any of these places, it will display a Choose Folder dialog to allow you to locate the file manually.

You can navigate the list of potential matches by using the up and down arrow keys or the mouse pointer, and open any listed file by selecting it and typing Return or Enter, or clicking the Open button.

If TextWrangler does not locate any potential matches, you can still search for the file, as before. (The search will skip locations where such a file would have already been found, i.e. the current file’s parent directory.)

If you type a string which appears to be an absolute or a home-relative path (e.g. “/path/to/some/file.txt” or “~/Documents/some/file.txt”, TextWrangler will cease searching and when you type Return or Enter, or click the “Open” button, TextWrangler will attempt to open the file at that path, if it exists.

If you type a string which appears to be a URL, TextWrangler will attempt to open it directly, or hand it off to an application that can. (TextWrangler supports a number of schemes, including ‘file’, ‘http’, ‘ftp’, and ‘sftp’.)

If you type in an unqualified partial path, e.g. “sys/errno.h”, TextWrangler will check the path components and only display files whose immediate ancestry matches what you entered. In this example, it would list “/usr/include/sys/errno.h” but not “/usr/include/errno.h”.

If you type in an absolute path, or a home-directory-relative path, e.g. “/usr/include/errno.h” or “~/.bash_profile”, TextWrangler will show the file if it exists at that location.

TextWrangler also maintains a search history in the Open File by Name window: when you open a matched item, TextWrangler will store the string you used, and the search history (magnifying glass) popup lists these recently used strings.

You may use the following wildcards as part of a search string:

Wildcard	Meaning
?	Any single character

Wildcard	Meaning
*	Any number of characters
#	Any numeric character
\	Escapes one of the above; for example, \? enters a question mark. To enter a literal backslash, use \\.

Using the Open Counterpart Command

You can use this command or its default key equivalent of Command-Option-uparrow (configurable via the Menus & Shortcuts preference panel) to switch between counterpart files (from source to header and vice versa). In addition to intrinsic counterparts (e.g. C/C++ style header/source mapping), you can explicitly define counterparts by setting a value for the (BBEdit- and TextWrangler-specific) "x-counterpart" variable in a file's Emacs variables. For example, if your file contains the following as part of its variable block:

```
-*- x-counterpart: ExampleStrings.R; -*-
```

when you type Command-Option-uparrow, TextWrangler will look for the file "ExampleStrings.R".

Using the Open Recent Command

The Open Recent submenu contains a list of files you have opened recently. To open one of these files, choose it from the Open Recent submenu. To set the number of items displayed in the Open Recent list, use the "Remember the [ ] most recently used items" option on the Application preference panel.

Using the Reopen using Encoding Command

The Reopen using Encoding submenu contains a list of all available text encodings. To reopen the current text document and have its contents interpreted using a different encoding, choose the desired encoding from the Reopen using Encoding submenu. This command will only be available if the current document is unmodified.

Quitting TextWrangler

By default, whenever you quit TextWrangler or TextWrangler automatically quits because of a system shutdown, restart, or user account logout, TextWrangler will attempt to restore as much of its state as possible when starting back up. Thus, you may not be prompted to save new or unsaved documents, since TextWrangler will automatically preserve the contents of all open documents before it exits.

You can control whether TextWrangler should preserve and restore unsaved changes via the "Restore unsaved changes" option in the Application preference panel existing documents. If this option is on, TextWrangler will automatically preserve unsaved changes. If this option is off, TextWrangler will instead prompt you to save each document which has unsaved changes.

An International Text Primer

Mac OS X includes extensive support for working with international text, including Unicode. If you have enabled additional text input methods in the International section of the System Preferences, you will see the Input menu on the right-hand side of the menu bar. This menu allows you to change keyboard layouts or script systems as you work.

Note Actually, even if you have never used a non-Roman script system before, you may still have used this menu, if you have ever chosen an alternate keyboard layout such as Dvorak, or a keyboard layout for a Roman language such as French. However, since the Roman script is suitable for several languages, choosing one of these keyboard layouts still leaves you in the Roman script.

International Text in TextWrangler

As a text editor, TextWrangler supports only one font per document window, though it can display all available characters in the active font, including Unicode characters.

TextWrangler supports editing in almost any language which uses left-to-right text input methods. To start entering text in any supported language, choose a suitable input method from the Input menu. The icon for that method will appear in the menu bar in place of either the American flag (for the U.S. English layout) or the icon for your usual Roman keyboard layout.

If you have turned off the "Try to match keyboard with text" option in the Options dialog of the International section of the System Preferences), you may also need to select a suitable display font via the Font panel. (We recommend leaving this option on, so that TextWrangler can automatically switch to the correct input method when you change document windows.)

You can use international text throughout TextWrangler—for example, in the Find window, within your text filters and scripts, or anywhere else you would use Roman text. Likewise, TextWrangler will provide the necessary style information so that if you copy and paste, or drag and drop, international text into another application, that application will have enough information to handle the text correctly (assuming it is capable of doing so).

TextWrangler remembers the encoding used in a document when you save it, so the next time you open it, you will not need to choose the font. However, you may not be able to read files which do not have this stored information, for instance, files downloaded from the Internet, until you choose an appropriate encoding for them.

When performing a search, TextWrangler respects any available information about each file's encoding. If a file does not contain any information about its encoding, TextWrangler will use the default encoding set in the Text Encodings Preferences panel.

Unicode

Unicode is an international standard for character encoding, which includes an extensive selection of characters from Roman, Cyrillic, Asian, Middle Eastern, and various other scripts. For more background information or complete details on Unicode, the Unicode Consortium web site is the best place to look.

```
http://www.unicode.org/
```

TextWrangler fully supports and makes extensive use of Unicode, in addition to all other OS-supported text encodings. In particular, TextWrangler internally represents all documents as Unicode, regardless of their on-disk encoding.

Saving Unicode Files

TextWrangler lets you save documents that use character set encodings other than Mac Roman, even multi-byte character sets. When saving a file, you can choose to save text composed in any script with any encoding. In addition to the standard character set encodings, TextWrangler also lets you save the files in a variety of plain Unicode files:

- **Unicode (UTF-8)**: UTF-8 **without** a byte-order mark
- **Unicode (UTF-8, with BOM)**: UTF-8 **with** a byte-order mark (BOM)
- UTF-16 Little-Endian
- UTF-16 Little-Endian, no BOM
- UTF-16
- UTF-16, no BOM

IMPORTANT The naming convention TextWrangler follows for UTF-8 documents has changed from that used by versions before 3.5: the encoding name "Unicode (UTF-8)" now refers to files **without** a byte-order mark (BOM), while the specific name "Unicode (UTF-8, with BOM)" refers to files which have a BOM.

Here are details about what each of the above options means:

- UTF-8: UTF-8 encoding is a more compact variant of Unicode that uses 8-bit tokens where possible to encode frequently used sequences from the file. (This format makes it easier to view and edit content in non-Unicode-aware editors.)
- UTF-16: UTF-16 encoding always uses 16-bit tokens.
- BOM: When saving Unicode files, you may include a byte-order mark (BOM) so that the reading application knows what byte order the file's data is in. However, since many applications do not correctly handle files which contain BOMs, you may wish to use an encoding variant without a BOM for maximum compatibility. (For purposes of recognition when you use this option, the UTF-16 BOM is FEFF, and the UTF-8 BOM is EFBBBF.)
- Little-Endian: Since UTF-16 uses two bytes to represent each character, this leaves the question of which of the two bytes comes first—whether it is "little-endian" or "big-endian." By default, TextWrangler writes UTF-16 big-endian (the standard). By choosing one of the "Little-Endian" (or "byte-swapped") encodings, you can write little-endian files instead, which some Windows software requires.

Opening Unicode Files

When opening files, TextWrangler will ordinarily determine the format of a file based on its file type and content, and automatically process Macintosh text, Unicode, and UTF-8. However, some files are structured such that TextWrangler is unable to correctly determine their format based on their type or contents. The cases that we know of are:

- UTF-8 files which lack a byte-order mark and do not contain any encoding specification **or** any extended characters. (If a UTF-8 has a byte-order mark, TextWrangler will correctly interpret its contents as UTF-8.)
- Byte-swapped Unicode files which were written without a byte-order mark (usually by broken Windows software);
- Unicode files which lack a byte-order mark.

If you know that a file you are trying to open is in Unicode but it displays as gibberish on your screen, close its window without saving. Then try reopening the file, using the Open As pop-up menu in the Open dialog to specify whether to treat the file as Unicode, byte-swapped (little-endian) Unicode, or UTF-8.

If you attempt to open a document which cannot be represented by either its declared encoding or any recognizable encoding, TextWrangler will present an alert to warn you. Also, if TextWrangler encounters such a file during a multi-file search, it will log a warning.

Accessing FTP/SFTP Servers

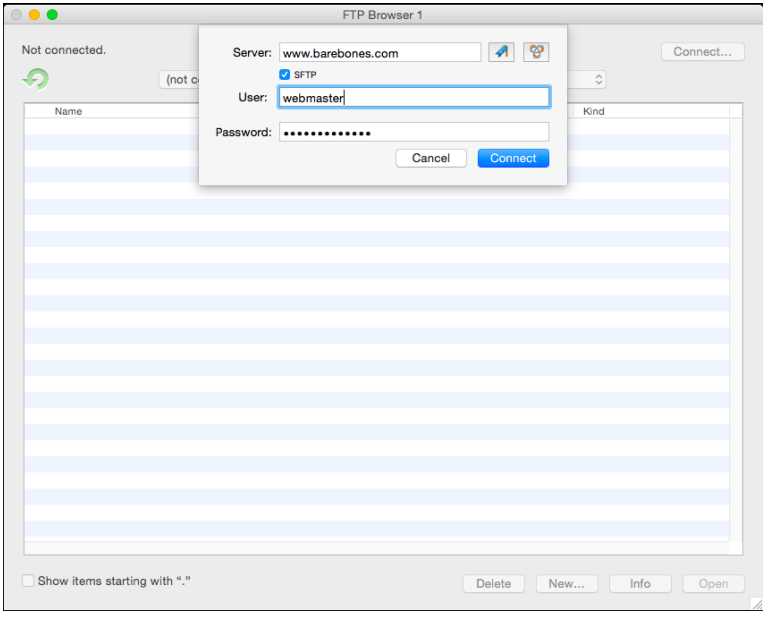
TextWrangler can open files directly from, and save them to, any available FTP server. It can also open and save files directly via SFTP (**S**SH **F**ile **T**ransfer **P**rotocol). In order to access a server via SFTP, that server must be running a compatible version of sshd. (A great many machines, including Mac OS X systems for which "Remote Login" is turned on in the Sharing panel of System Preferences, satisfy these criteria.)

Aside from choosing the SFTP checkbox in the Open from…/Save to… dialogs, or the FTP/SFTP Browser, opening and saving files via SFTP works just like it does when using ordinary FTP. A file opened via SFTP will appear in the Open Recent submenu with an "sftp:" URL, and you can send a "get URL" event to TextWrangler with an "sftp" URL as well.

Opening Files from FTP/SFTP Servers

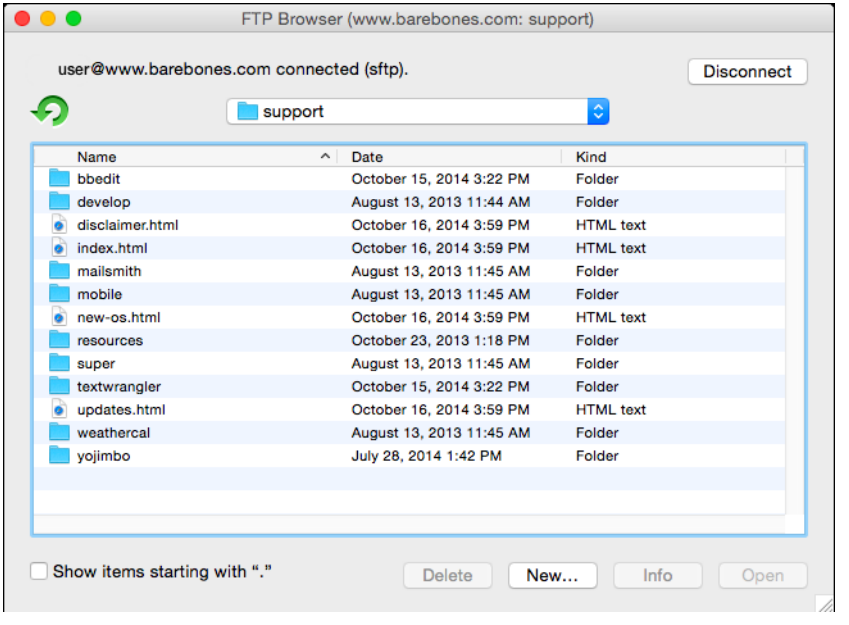
To directly open files from an FTP or SFTP server, choose Open from FTP/SFTP Server from the File menu. TextWrangler will open an new FTP/SFTP Browser window. Like other browser windows, FTP/SFTP browsers will remain open until you close them, and once connected, they will maintain a persistent connection to the server for as long as they remain open

Enter the server's name in the "Server:" field, or choose a local server advertised by Bonjour by clicking the popup menu to the right of the "Server:" field, specify your user name and password in the appropriate fields and choose the "SFTP" option if appropriate; then click the Connect button or press the Return or Enter keys to connect to the server.



Alternatively, you can choose a bookmark from the Bookmarks pop-up menu to fill in stored info for the server, user name, password, and connection options. You can choose Add Bookmark... from the directory popup in an FTP/SFTP Browser window to create a bookmark for the current server and directory, or create arbitrary bookmarks via the Bookmarks panel of the Setup window. You can modify or delete existing bookmarks via the Bookmarks panel of the Setup window.

Once you have connected to the server, you can open files by double-clicking them, or selecting them and clicking the Open button. You can double-click a folder to change directories. If you hold down the Option key when opening a folder, it will open in a new FTP/SFTP Browser window. You can select a range of files and directories by Shift-clicking, and you can select (and deselect) multiple items one at a time by Command-clicking them.


FTP Browser (www.barebones.com: support)
user@www.barebones.com connected (sftp).
Disconnect
support
Name
Date
Kind
bbedit October 15, 2014 3:22 PM Folder
develop August 13, 2013 11:44 AM Folder
disclaimer.html October 16, 2014 3:59 PM HTML text
index.html October 16, 2014 3:59 PM HTML text
mailsmith August 13, 2013 11:45 AM Folder
mobile August 13, 2013 11:45 AM Folder
new-os.html October 16, 2014 3:59 PM HTML text
resources October 23, 2013 1:18 PM Folder
super August 13, 2013 11:45 AM Folder
textwrangler October 15, 2014 3:22 PM Folder
updates.html October 16, 2014 3:59 PM HTML text
weathercal August 13, 2013 11:45 AM Folder
yojimbo July 28, 2014 1:42 PM Folder
Show items starting with "."
Delete
New...
Info
Open

To refresh the directory listing, click the button with the circular arrow icon (located above the upper left corner of the listing). The checkbox below the listing labeled Show Files Starting with "." tells TextWrangler whether to display hidden or admin files in the chosen directory, such as .login, .forward, and .signature. (Starting a file name with a period is a convention used by Mac OS X and other Unix systems to make that file invisible in most directory listings.)

Once you have selected a file and opened it, TextWrangler displays the file in a text editing window. The toolbar displays the URL of the file on the server, not the pathname of the file on your hard drive as it does for local files.

NEW You can drag items from FTP browser windows to other applications. TextWrangler will include a URL in the drag event for each selected item in a form that applications which accept URLs may be able to use.

You can use the Info button to examine the size, modification date, and if applicable, file system permissions of the selected file. You can edit the file's name and click the Rename button to rename the file on the server; you can also make changes to the permissions and click the Set button to change them. (Take care not to set the permissions such that the file becomes inaccessible to you!)

You can directly create a new file (or folder) on the server by clicking the New button, or remove files from the server by selecting them and pressing the Delete button.

Specifying Alternate Ports

TextWrangler allows you to open an FTP or SFTP connection on ports other than the default. To specify an alternate port, place it at the end of the server name, separated by a colon—for example, ftp.example.com:1111.

Storing Passwords

As long as your user account's keychain is unlocked, TextWrangler will use it to store the password for each server that you access, and to automatically fill in the corresponding password whenever you enter a server and user name pair for which there is a keychain entry. If your keychain is locked, you will need to retype your password every time you use the FTP browser.

Using SSH Key Files

In order to connect to an SFTP server which requires SSH keys (or certificates) rather than passwords, you must first create an appropriate entry for that server in your local account's .ssh/config. You may then type the server name, or shortcut name, into the Server field of the FTP/SFTP Browser and connect without entering a password.

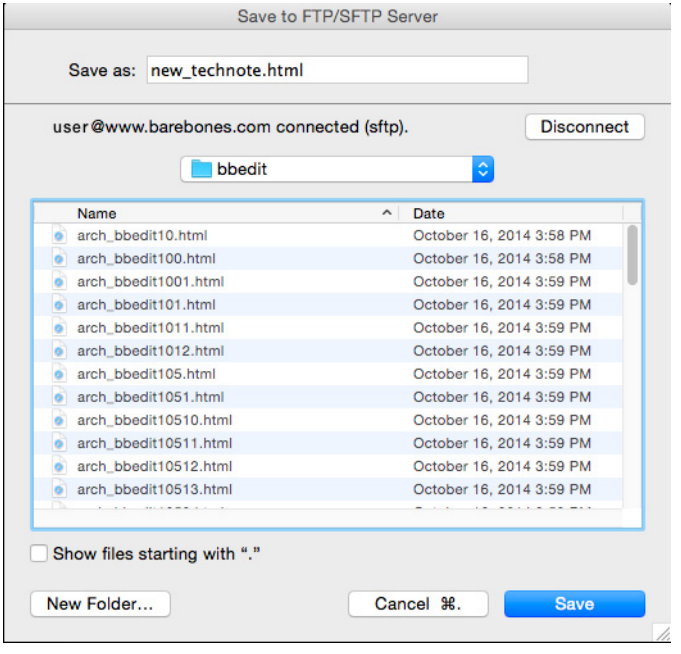
Transfer Formats

When you open a file from an FTP or SFTP server, TextWrangler downloads the file "raw" (in binary mode) and then performs a standard line ending conversion upon opening the (local temp) file.

Saving Files to FTP/SFTP Servers

After you have edited a file opened from an FTP or SFTP server, pressing Command-S or choosing Save from the File menu saves the new version back to the server. If you want to save the file in a different directory or under another name, choose Save to FTP/SFTP Server to open the Save to FTP/SFTP Server dialog (shown below).

This dialog works much like the standard Save dialog for saving a local file, with the addition of fields and controls similar to those in the FTP/SFTP browser allowing you to select or specify connection info, and to navigate and obtain info about other files.


Save to FTP/SFTP Server
Save as: new_technote.html
user@www.barebones.com connected (sftp).
Disconnect
bbedit
Name
Date
arch_bbedit10.html October 16, 2014 3:58 PM
arch_bbedit100.html October 16, 2014 3:58 PM
arch_bbedit1001.html October 16, 2014 3:59 PM
arch_bbedit101.html October 16, 2014 3:59 PM
arch_bbedit1011.html October 16, 2014 3:59 PM
arch_bbedit1012.html October 16, 2014 3:59 PM
arch_bbedit105.html October 16, 2014 3:59 PM
arch_bbedit1051.html October 16, 2014 3:59 PM
arch_bbedit10510.html October 16, 2014 3:59 PM
arch_bbedit10511.html October 16, 2014 3:59 PM
arch_bbedit10512.html October 16, 2014 3:59 PM
arch_bbedit10513.html October 16, 2014 3:59 PM
Show files starting with "."
New Folder...
Cancel ⌘.
Save

Note When you save a file to an FTP or SFTP server using either Save or Save to FTP/SFTP Server, and the file has Unix (LF) or Windows (CR+LF) line endings, TextWrangler uploads the file in binary mode, preserving its line endings exactly as they are on your local machine. However, if the file has Macintosh (CR) line endings, it is uploaded in text mode so that the server can convert the line endings as appropriate.

Finally, you can use Save a Copy to FTP/SFTP Server to upload a copy of your current file to an FTP server while keeping your local file open. This is especially useful when you maintain web site content on your local hard drive and only need to upload changes made in one or two files to the server.

Using TextWrangler from the Command Line

You can use the “edit” command line tool to open files into TextWrangler via the Unix command line. The first time you run TextWrangler after installation, it will offer to install the command line tools for you. If you choose not to do so, you can choose “Install Command Line Tools” from the TextWrangler (application) menu at any time to install (or re-install) the current version of the command line tools.

To open a file in TextWrangler from the command line, type

```
edit filename
```

where *filename* is the name of the file to be opened. To launch TextWrangler without opening a file (or activate it, if it is already running), type

```
edit -l
```

In addition to files, you can also specify FTP or SFTP URLs to files or directories, to have TextWrangler open the specified files, or an FTP/SFTP Browser for each directory. You will be prompted to enter passwords if necessary.

You can also pipe STDIN to the “edit” tool, and it will open in a new untitled window in TextWrangler: for example,

```
ls -la | edit
```

If you just type

```
edit
```

with no parameters, the tool will accept STDIN from the terminal; type Control-D (end-of-file) to terminate and send it to TextWrangler.

The complete command line syntax for the “edit” tool is

```
edit [ -bcChlpsuvVw --(long_form_switches) ]
 [ -e <encoding_name> ] [ -t <string> ]  [ +<n> ]
 [ file (or) <S/FTP URL> ... ]
```

See the “edit” tool’s man page (‘man edit’) for a complete description of the available switches and options.

Using Stationery

Like most Macintosh applications, TextWrangler supports stationery pads. A stationery pad is a template file that, when opened, results in a new, untitled document with the content from the original stationery file. In other words, you do not edit the stationery file itself; you use it as a starting point for a new document.

To create a stationery pad, click the Save As Stationery checkbox when saving the file from TextWrangler. Alternatively, you can change any document into a stationery pad in the Finder by clicking the Stationery Pad checkbox in the document’s Get Info window.

You can create new documents from a stationery pad in any of these ways:

- Open the pad the same way you would open any other document.
- Choose New With Stationery from the File menu, and select the desired stationery pad from the contents of the Stationery folder (inside TextWrangler's application support folder).

You can also assign a keyboard shortcut to a stationery file in the Menus & Shortcuts preference panel (in the File -> New with Stationery section).

Manually Sorting the Stationery List

By default, items in the Stationery List are displayed in alphabetical order. However, you can force them to appear in any desired order by including any two characters followed by a right parenthesis at the beginning of their names. For example, "00)Markdown templates" would sort before "01)HTML Templates". For such files, the first three characters are not displayed in TextWrangler. You can also insert a divider by including an empty folder whose name ends with the string "-***". (The folder can be named anything, so it sorts where you want it.)

Hex Dump for Files and Documents

Choose the Hex Dump File command to generate a hex dump representation from a file that you choose. Choose Hex Dump Front Document to generate a hex dump representation of the frontmost document as it exists in memory.

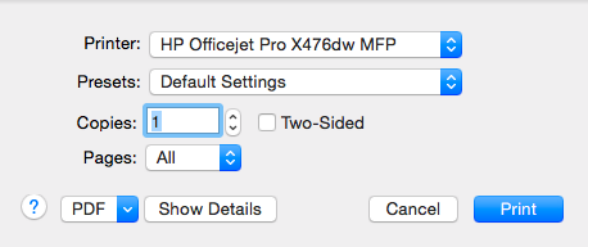
You should bear in mind that the result of performing the Hex Dump command against a disk file may differ from the result obtained by using it against an open document, since when a document is open in memory, even without any explicit edits being made, line-break translation and possibly character set encoding conversions have taken place.

Making Backups

TextWrangler can automatically make a backup copy of each document you edit before saving it. To enable this feature, turn on the "Make backup before saving" option in the Text Files preference panel. For complete details on how this feature works, and optional behaviors, please see "Make backup before saving" on page 193.

Printing

To print a document, choose the Print command from the File menu. TextWrangler will display a standard print sheet in that document's window.

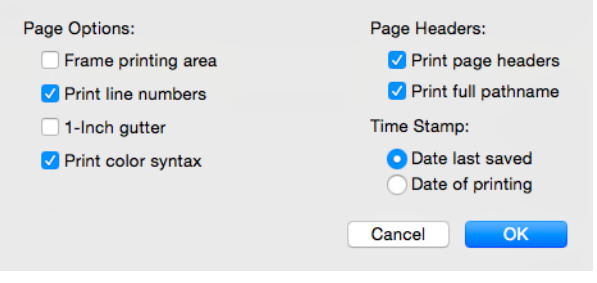


To print one copy of the active document without displaying the print sheet, hold down the Option and Shift keys and choose the Print One Copy command from the File menu.

To print only the selected range of text within the active document, choose the Print Selection command from the File menu.

Printing Options

You can access TextWrangler's application-specific printing options for the current document by choosing the Printing Options command in the Edit menu. When you choose this command, TextWrangler will open the printing options sheet.



Note You can set defaults for these options, as well as the printing font, in the Printing panel of TextWrangler's Preferences window.

Page Options:

These options control how the printed pages will be laid out.

Frame printing area

When this option is selected, TextWrangler draws a frame around the printed text.

Print line numbers

When this option is selected, TextWrangler prints line numbers along the left edge of the paper.

1-Inch gutter
When this option is selected, TextWrangler leaves a one-inch margin along the left edge of the paper. Use this option if you usually put your pages in three-ring binders.

Print color syntax
When this option is selected, TextWrangler will print the document in color.

Page Headers:
These options control what information is included in the page headers.

Print page headers
When this option is selected, TextWrangler prints the page number, the name of the file, and the time and date printed in a header at the top of each page.

Print full pathname
When this option is selected, TextWrangler prints the full pathname of the file in the header.

Time Stamp
The Time Stamp option lets you choose whether the date that appears in the header is the date that the file was last modified or the date that the file was printed.

CHAPTER

4 Editing Text with TextWrangler

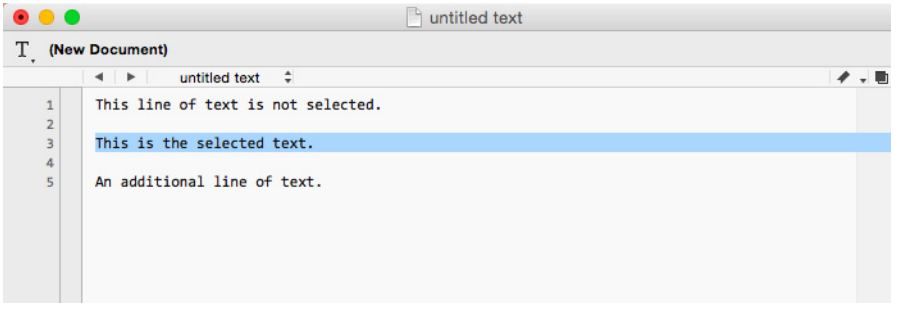
This chapter describes the basics of editing text with TextWrangler, wrapping text, text manipulations, and file comparison.

In this chapter

Basic Editing **60**
Moving Text – 60 • *Multiple Clipboards* – 61
Drag and Drop – 61
Multiple Undo **62**
Window Anatomy **63**
The Toolbar – 63 • *The Split Bar* – 65 • *The Navigation Bar* – 65
The File List – 69 • *The Status Bar* – 71
The Gutter and Folded Text Regions – 73
The View Menu **75**
Text Display – 75
Cursor Movement and Text Selection **78**
Clicking and Dragging – 78 • *Arrow Keys* – 79
CamelCase Navigation – 79 • *Rectangular Selections* – 79
Working with Rectangular Selections – 80
Scrolling the View – 83 • *The Delete Key* – 83
The Numeric Keypad – 83 • *Line Number Command* – 84
Function Keys – 84 • *Resolving URLs* – 85
Text Options **85**
Editing Options – 86
Text Options **85**
Editing Options – 86 • *Display Options* – 86
How TextWrangler Wraps Text **88**
Soft Wrapping – 89
Hard Wrapping – 90
The Insert Submenu **93**
Inserting File Contents – 93 • *Inserting File & Folder Paths* – 93
Inserting a Folder Listing – 93 • *Inserting a Page Break* – 93
Comparing Text Files – 94
Comparing Text Files **94**
Compare Against Disk File – 96
Multi-File Compare Options – 97
Using Markers **98**
Setting Markers – 99 • *Clearing Markers* – 99
Using Grep to Set Markers – 99
Spell Checking Documents **100**
Check Spelling As You Type – 100 • *Manual Spell Checking* – 101
The Spelling Panel – 101

Basic Editing

TextWrangler behaves like most Macintosh word processors and text editors. Characters that you type in an active window appear at the insertion point, a vertical blinking bar. You can click and drag the mouse to select several characters or words, and the selected text is highlighted using the system highlight color, which you can set in the Appearance panel of the System Preferences.



If you select some text and then type, whatever you type replaces the selected text.

To delete selected text, press the Delete key or choose Clear from the Edit menu. If you have a keyboard with a numeric keypad on it, you can press the Clear key on the keypad to delete the selected text.

In addition to clicking and dragging to select text, you can use the selection commands in the Edit menu.

To select…	Choose this from the Edit menu…
All text	Select All
No text (deselect)	(click anywhere in the document, or type any arrow key)
Line containing insertion point	Select Line
Paragraph containing insertion point	Select Paragraph

You can then cut, copy, or perform any other action that affects the selected text.

Note TextWrangler defines a paragraph as a block of text surrounded by blank lines (lines containing no characters other than tabs or spaces). The beginning and end of the document also mark the beginning and end of paragraphs.

Moving Text

To move text from one place to another, follow these steps:

1 **Select the text you want to move.**

2 **Choose Cut from the Edit menu.**

TextWrangler removes the text from the window and stores it on the clipboard.

3 **Use the scroll bars to move to the new place for the text if necessary; then click to set the insertion point where the text is to be inserted.**

4 **Choose Paste from the Edit menu.**

You can paste the contents of the clipboard as many times as you want in any TextWrangler window or in any other application.

Pasting inserts the text stored on the clipboard at the insertion point. If there is a selection, pasting replaces the selection with the contents of the clipboard.

To place text on the clipboard without deleting it, choose Copy from the Edit menu.

Tip To add selected text to the existing contents of the clipboard, hold down the Shift key as you choose the Cut or Copy command. When you hold down the Shift key, TextWrangler changes these commands to Cut & Append and Copy & Append.

Multiple Clipboards

TextWrangler supports six separate clipboards. Each time you use the Cut or Copy command, TextWrangler automatically switches to the next clipboard (wrapping back around to the first clipboard after the sixth). This way, the last six things you copied or cut are always available for pasting—sort of a "clipboard history."

By default, the Paste command pastes text from the most recently used clipboard, so if you do nothing special, TextWrangler works just like any other Macintosh program. However, by using the Previous Clipboard command in the Edit menu you can access the previous clipboard contents. Next Clipboard moves forward through the clipboard history.

Once you have selected a clipboard using one of these methods, the next Cut, Copy, or Paste command will use the clipboard you chose. (Subsequent Cut or Copy commands will advance to the next clipboard; Paste never advances automatically.)

Holding down the Shift key changes the Paste command to Paste Previous Clipboard, or you can use the key equivalent Command-Shift-V. This command replaces the pasted text with the contents of the previous clipboard. The previous clipboard becomes current and will be used for any further paste operations; repeated applications of the command cycle backward through the available clipboards.

Drag and Drop

Another way to move text from one place to another is by "drag and drop." If you drag and drop text from one window to another, TextWrangler copies the text to the target window without removing it from the original window.

In addition, you can drag and drop an item from the Finder onto an editing window in TextWrangler. If the item is a text file, the file's contents are inserted. If the item is a folder, a listing of the item's contents is inserted. If you hold down the Command key while dragging a folder, the path of the item is inserted instead.

Multiple Undo

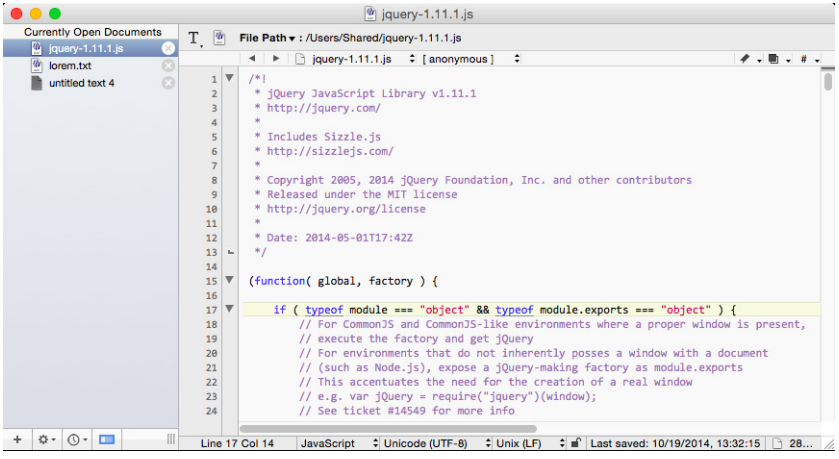
TextWrangler provides the ability to undo multiple edits, one action at a time. The number of edits that may be undone is limited only by available memory. The practical limitation is determined by the extent of the edits and the amount of free memory.

TextWrangler also supports multiple Redos. If you have not made any changes after performing an Undo, you can redo each action, in order, by choosing that Redo command from the Edit menu or typing Command-Shift-Z. However, once you perform a new action, you cannot redo any actions that you undid before you made that change.

There is also a Clear Undo History menu command (Command-Control-Z), which will clear the undo history for the current editing window. This command can be useful if you have performed many operations on a file and wish to recover memory stored by Undo state information (in the rare event that should become necessary). You can also script this operation via the "clear undo history" scripting command (see the scripting dictionary for details).

Window Anatomy

TextWrangler text windows have the same controls you are familiar with from other Macintosh applications (for example, text windows are resizable and zoomable, and have both vertical and horizontal scroll bars). Some additional elements which may be less familiar are the toolbar, the split bar, the navigation bar, and the file list.


jquery-1.11.1.js
Currently Open Documents
jquery-1.11.1.js
lorem.txt
untitled text 4
File Path : /Users/Shared/jquery-1.11.1.js
jquery-1.11.1.js
[anonymous]
/*!
* jQuery JavaScript Library v1.11.1
* http://jquery.com/
*
* Includes Sizzle.js
* http://sizzlejs.com/
*
* Copyright 2005, 2014 jQuery Foundation, Inc. and other contributors
* Released under the MIT license
* http://jquery.org/license
*
* Date: 2014-05-01T17:42Z
*/
(function(global, factory) {
if (typeof module === "object" && typeof module.exports === "object") {
// For CommonJS and CommonJS-like environments where a proper window is present,
// execute the factory and get jQuery
// For environments that do not inherently posses a window with a document
// (such as Node.js), expose a jQuery-making factory as module.exports
// This accentuates the need for the creation of a real window
// e.g. var jQuery = require("jquery")(window);
// See ticket #14549 for more info
Line 17 Col 14
JavaScript
Unicode (UTF-8)
Unix (LF)
Last saved: 10/19/2014, 13:32:15

Under OS X 10.8 (Mountain Lion) and OS X 10.9 (Mavericks), you can expand the current editing window into full screen mode by clicking the double-arrow control in the top-right corner of the window.

Under OS X 10.10 (Yosemite), you can instead click the green “Zoom” button to expand the current editing window into full screen mode.

IMPORTANT You can choose whether TextWrangler should display all new and opened documents in the frontmost window, or open each document into a new text window, by setting the “Open documents into the front window when possible” option in the Application preference panel (see page 180).

The Toolbar

The toolbar is a section at the top of each editing window which contains a popup menu (the Text Options popup) you can use to adjust display options, a proxy icon for the current document, and the current document’s path on disk (if applicable). You can toggle display of the toolbar by choosing Hide Toolbar/Show Toolbar in the View menu.

You can also change the Toolbar options in the Appearance preference panel to make TextWrangler hide or show individual items on the toolbar by default.

If the current document has a corresponding disk file, the toolbar displays the full path to the document’s disk file and the last time the file was saved. If the document has not been saved to disk, the toolbar displays “(New Document)” instead of a file name.

Note Windows in which the toolbar is not directly below the window title bar (for example, disk browsers and search results) do not have a toolbar control, but do honor the global toolbar preference. You can also use the Text Options sheet to show and hide the toolbar on a per-window basis.

The icons on the toolbar are indicators, buttons, and popup menus that give you quick access to commonly used functions. The following table explains each icon.

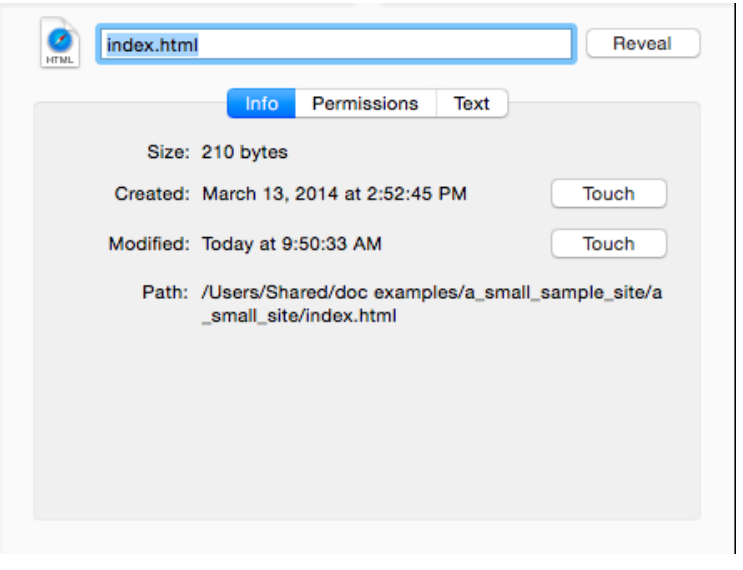
Icon	Meaning
	The Text Options popup menu contains commands such as Soft Wrap Text, Show Page Guide, and Show Invisibles that let you control how text is displayed in the window.
	The document proxy icon represents the current document. Clicking this icon will open the document information panel; see the following section for further details. Command-clicking on this icon has the same effect as choosing Reveal in Finder from the View menu: it opens a Finder window that contains the document. You can also drag the document proxy icon to any other application, or you can drag it to the Trash (which is the same as choosing Close & Delete from the File menu).

Key Equivalents for Toolbar Menu Items

You can assign keyboard shortcuts to items on the Text Options popup menu from the Toolbar entry in the Menus & Shortcuts preference panel.

The Document Information Panel

Clicking on the document icon in a window's toolbar will now open a spring-loaded info panel which displays basic information about the file, and gives you the ability (for local files only) to rename it, "touch" its creation/modification dates, and change its permissions. To dismiss this panel, click outside it, switch to another window or application, or press the Escape key.

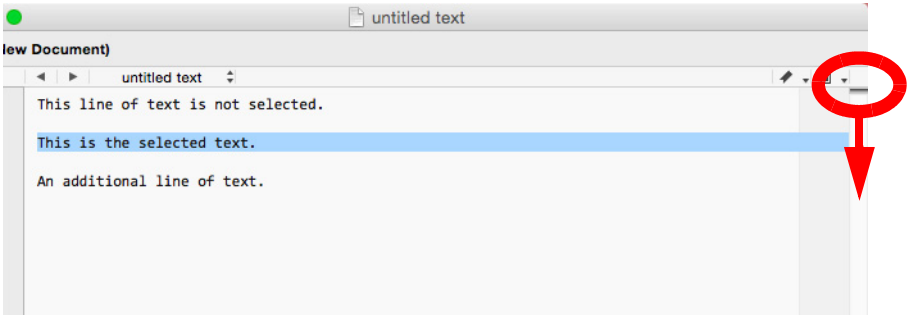


The Split Bar

Every text window and every browser text pane has a split bar, a small black bar above the scroll bar, that lets you split it into two active view regions. Splitting a text pane lets you view and edit a document's content in two places at the same time. Each region is independently scrollable.

Note Scrolling the non-active split region does not automatically change view focus.

To split the text pane, simply drag the split bar down and let go.



To collapse the text pane back down to a single region, drag the split bar (starting from anywhere along its length, not just at its right end) back up to its original position.

Tip Double-clicking the split bar unsplits a split text pane or restores the last-used split position. If the text pane has never been split, it will be split 50-50. To force a 50-50 split for a previously split text pane, Option-double-click the split bar when it is in its original position.

The Navigation Bar

The navigation bar is a panel at the top of a text window which provides controls for selecting the active document and for moving to specific points with the current document. To hide the navigation bar, choose Hide Navigation Bar in the View menu, or turn off the Navigation Bar options in the Appearance preference panel.

You can also use the options in the Appearance preference panel to hide or show individual items on the navigation bar.

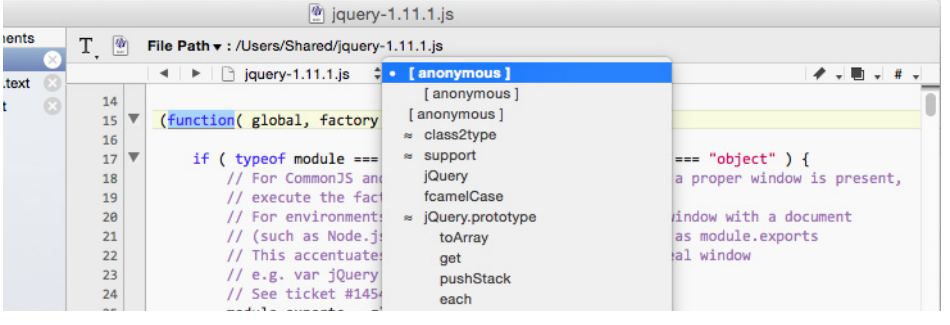
Choosing the Active Document

Click on any document in the file list to make that document active, click the Previous or Next buttons to move to the previous or next document in the window, or choose Previous Document/Next Document from the View menu. You can also choose a specific document from the adjacent popup menu to make it active.

The Previous and Next buttons in the Navigation bar, as well as the Previous Document/ Next Document commands, select documents in most-recently used order, rather than alphabetical order.

Function Navigation

The Function popup menu lists the functions defined in a programming language source file or various specific tags present within an HTML document. If the current document's language does not support function scanning, the function popup will not be displayed in the navigation bar.



The following indicators appear in the function popup to show the type of function.

Indicator	Meaning
•	The function containing the insertion point
†	C/C++ typedef
◊	C/C++ "#pragma mark" directive
italic name	C/C++ function prototype
1-6	Heading level (in HTML files)

Indicator	Meaning
tag name	Tag name for the indicated name or ID attribute value (in HTML files)

Manually Defined Functions

For code written in several languages, including C/C++, PHP, Python, and Ruby, you can manually add customized entries to the function popup menu by inserting suitable “mark” directives within a document.

In C/C++ documents, TextWrangler recognizes “#pragma mark” directives. For other languages, each directive consists of a line comment followed by a space and the string “#mark ”, plus the desired marker string.

The complete set of ‘mark’ directives includes:

```
mark, fixme, fix-me, note, nyi, review, todo, to-do, xxx, ???, !!!
```

For example, to add an function popup entry named “My item”:

In C/C++:

```
#pragma mark My item
```

In JavaScript:

```
// #mark My item
```

In HTML:

```
<!-- #mark My item -->
```

In PHP

```
// #mark My item
```

```
// #mark: My item
```

Note Callout markers may be uppercase, lowercase, or mixed case, and may have a single, optional colon at the end, before the space and the text to appear in the function popup.

In Python:

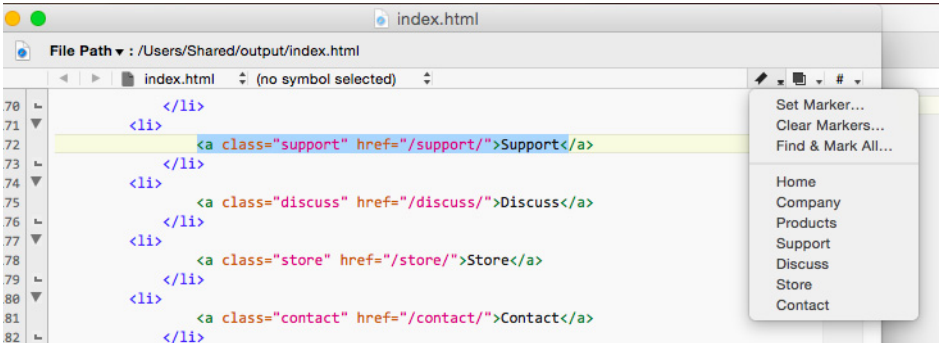
```
#mark My item
```

```
# #mark My item
```

Note In Python files, each directive must be separated from the preceding content by at least one empty line.

Navigation with Markers

A marker is a selection range that you can name. If a document contains any markers, you can select them from the Marker popup menu to move quickly to the specified section of the file.

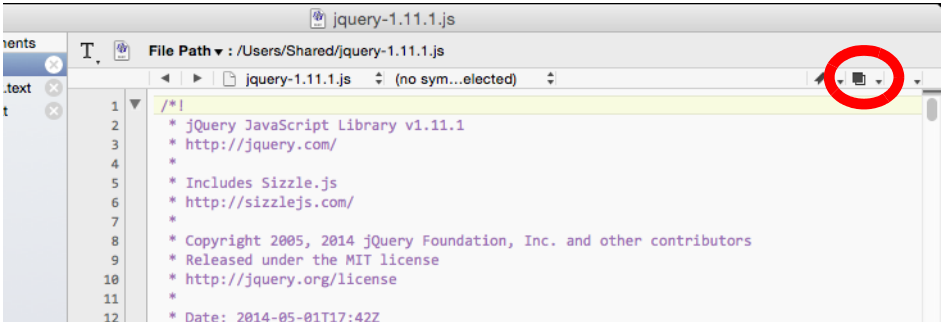


For more information on working with markers, please see “Using Markers” on page 98.

Note If you are programming, you may be tempted to use markers to mark functions in your source code. However, if TextWrangler supports the language you are using, this is usually unnecessary; your functions will automatically appear in the Function popup menu.

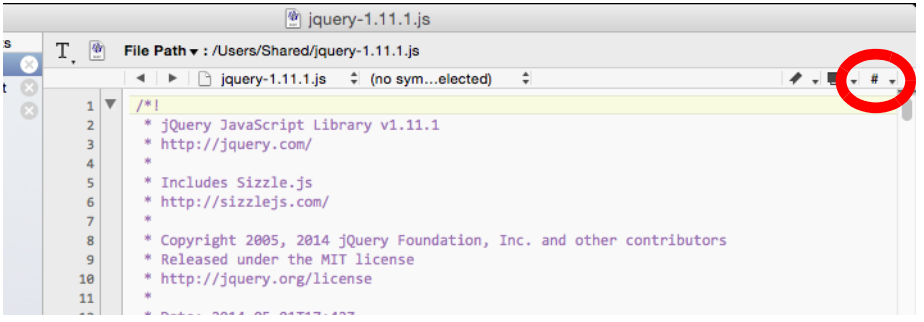
Opening Counterparts

You can press the Counterpart button next to the Marker popup to quickly open and/or switch back and forth between a file and its counterpart (source file to header, or vice versa). This button has the same effect as Open Counterpart in the File menu (see page 47).



Opening Included Files

You can use the Included Files popup to list or open any included files which the current document references.



Key Equivalents for Navigation Bar Menu Items

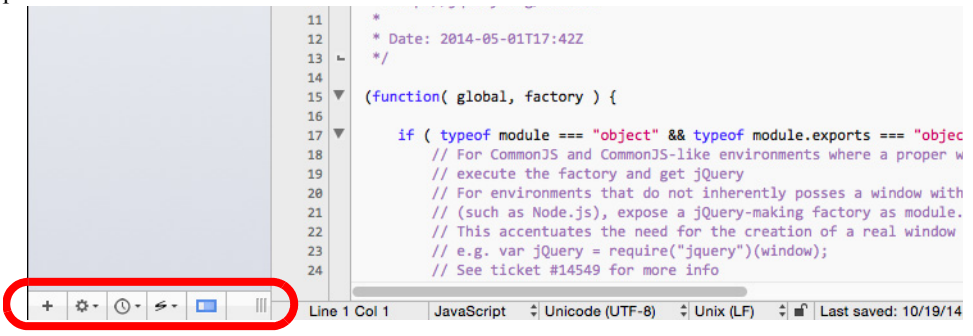
You can assign key equivalents to the controls on the navigation bar from the Navigation Bar entry in the Menus & Shortcuts preference panel. So, for example, you can assign a key equivalent to Open Function Menu, then press that key combination and use the arrow keys to navigate the current document's function list directly from the keyboard.

The File List

If TextWrangler is configured to open documents into the front window, it will display a file list down the left-hand side of each editing window which shows all the documents currently open in that window. To hide (or show) the file list, choose the Show Files (or Hide Files) command in the View menu, or type its default key equivalent of Command-0. Click any document's name in the list to make that document frontmost in the text window.

Dragging a document's name from the file list has the same effect as dragging its proxy icon in the toolbar. You can also drag documents within the list to manually reorder them

There are several buttons and popup menus below the file list, which you can apply to perform various additional actions.


* Date: 2014-05-01T17:42Z */
(function(global, factory) {
if (typeof module === "object" && typeof module.exports === "objec
// For CommonJS and CommonJS-like environments where a proper w
// execute the factory and get jQuery
// For environments that do not inherently posses a window with
// (such as Node.js), expose a jQuery-making factory as module.
// This accentuates the need for the creation of a real window
// e.g. var jQuery = require("jquery")(window);
// See ticket #14549 for more info
Line 1 Col 1
JavaScript
Unicode (UTF-8)
Unix (LF)
Last saved: 10/19/14

To open an existing file into the current text window, choose Open from the File menu, or drag and drop the file from the Finder into the window's file list.

To create a new document, click the Add (plus) button or choose New Text Document in the New submenu of the File menu.

To move a document from the current text window into its own text window, select it in the list and choose Move to New Window from the Action (gear) popup menu, or Control-click on the document in the list and choose this command from the contextual menu. To move multiple documents, select them and choose Move to New Window to create a new text window containing all the selected documents.

To close a document, you can choose Close Document from the File menu, click on the close box next to its name in the list, select it in the list and apply the Close command from the action (gear) menu, or Control-click on it in the list and select Close in the contextual menu. You can also choose the Close Others command from the action menu or in the contextual menu to close all documents except the selected document.

To reopen any recently open file, click on the Recent (clock) popup and select the desired file.

To move a document from one text window to another, drag its name from the first text window's file list into the second text window's file list. You can select and move multiple documents at once.

To save the current document, you can choose Save from the File menu or the action menu. To save multiple documents at once, select them and choose Save from the Action menu, or Control-click on them and select Save in the contextual menu. To save all documents in the window at once, hold down the Option key and choose Save All from the Action menu.

The Status Bar

The status bar is located directly to the left of the horizontal scrollbar. The status bar displays the current cursor position and contains popup menus showing the language, text encoding, and line break format of the current document. To hide the status bar, turn off the Show Status Bar option in the Appearance preference panel.

* http://jquery.com/
* Includes Sizzle.js
* http://sizzlejs.com/
* Copyright 2005, 2014 jQuery Foundation, Inc. and other contributors
* Released under the MIT license
* http://jquery.org/license
* Date: 2014-05-01T17:42Z
(function(global, factory) {
if (typeof module === "object" && typeof module.exports === "object") {
// For CommonJS and CommonJS-like environments where a proper window is present,
// execute the factory and get jQuery
// For environments that do not inherently posses a window with a document
// (such as Node.js), expose a jQuery-making factory as module.exports
// This accentuates the need for the creation of a real window
// e.g. var jQuery = require("jquery")(window);
// See ticket #14549 for more info
Line 1 Col 1
JavaScript
Unicode (UTF-8)
Unix (LF)
Last saved: 10/19/14, 1:32:15 PM

You can also use the options in the Application preference panel to hide or show individual items on the status bar.

Cursor Position

This section of the status bar shows the current line and character position of the insertion point.

Language

The Language popup menu displays the language mapping for the current document. You can change this mapping by choosing a different language from the popup.

Text Encoding

The Text Encoding popup menu displays the encoding used to open the current document. You can change the encoding in which the document will be saved by choosing a different encoding from the popup.

To choose an arbitrary encoding, even one not currently displayed, choose Other from the popup and pick your desired encoding from the resulting list.

Line Break Type

The Line Break Type popup menu shows the line break format of the current document's disk file. You can change the line break format with which the file will be saved by choosing it from the popup.

Document Lock State

The padlock icon immediately to the left of the Document Save Date indicates whether the document is currently writable or locked.

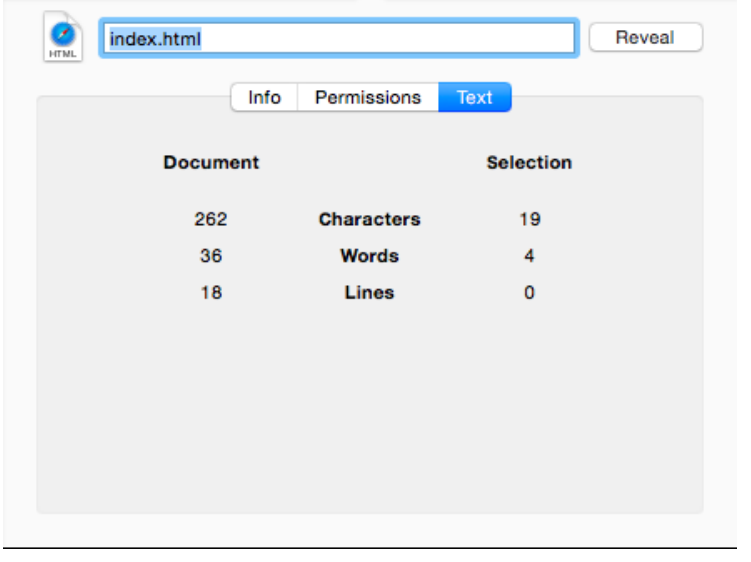
Document Save Date

The Document Save Date section of the status bar displays the date and time that the document was last saved (if applicable).

Document Statistics

This section of the status bar dynamically displays the number of characters, words, and lines in the document or the active selection (if any). (This icon will be white when TextWrangler is displaying statistics for the whole document, and green when it's displaying statistics for the current selection.)

You may also click on the statistics section at any time to display the Text tab of the document information panel. between displaying info for the whole document and for the selection range.



Magnification

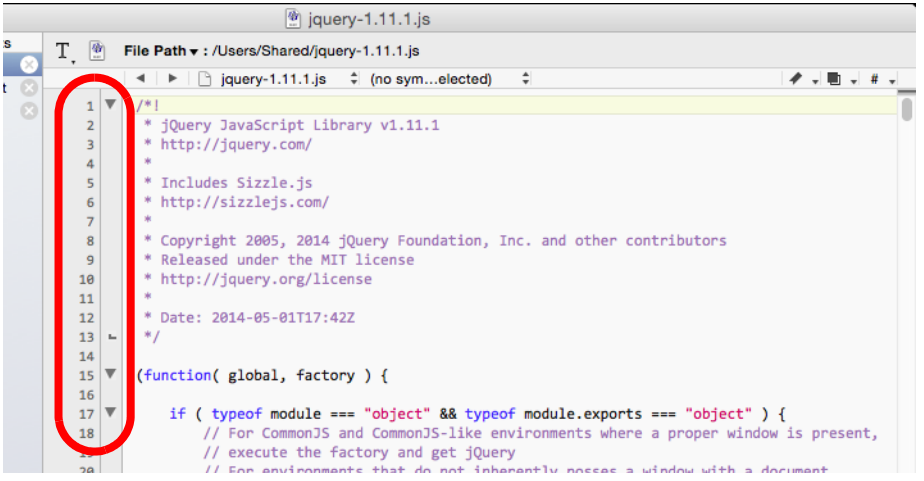
The Magnification popup menu displays the current document's magnification level and allows you to change it. The default magnification level is 100%.

Key Equivalents for Status Bar Items

You can assign key equivalents to the items on the status bar from the Status Bar entry in the Menus & Shortcuts preference panel. For example, you can assign a key equivalent to the Line Breaks popup, then press that key combination and use the arrow keys to select the desired line break option directly from the keyboard.

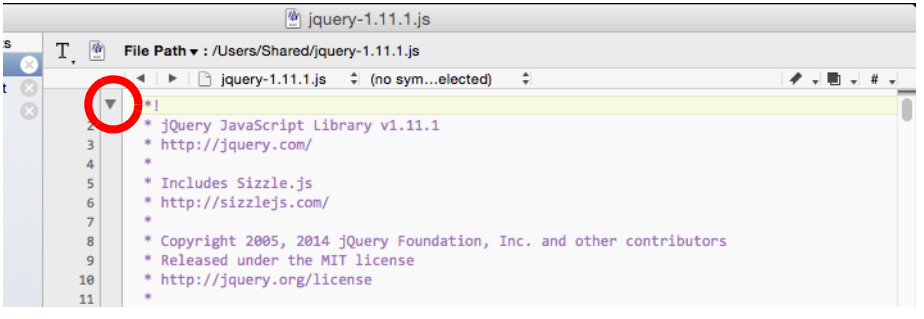
The Gutter and Folded Text Regions

The gutter is the vertical bar directly to the left of the text area, and immediately to the right of the line number display bar (not shown), which contains indicators for folded and foldable regions (automatically-generated folds).


jquery-1.11.1.js
File Path : /Users/Shared/jquery-1.11.1.js
jquery-1.11.1.js
(no sym...elected)
/*!
* jQuery JavaScript Library v1.11.1
* http://jquery.com/
*
* Includes Sizzle.js
* http://sizzlejs.com/
*
* Copyright 2005, 2014 jQuery Foundation, Inc. and other contributors
* Released under the MIT license
* http://jquery.org/license
*
* Date: 2014-05-01T17:42Z
*/
(function(global, factory) {
if (typeof module === "object" && typeof module.exports === "object") {
// For CommonJS and CommonJS-like environments where a proper window is present,
// execute the factory and get jQuery

Folding Controls

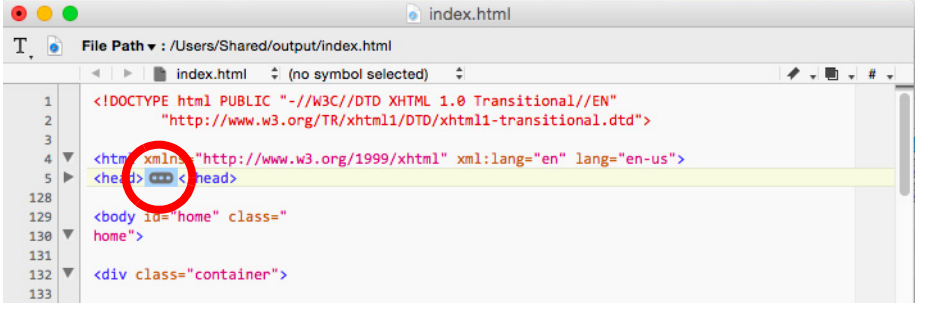
The triangular controls displayed in the gutter are disclosure triangles; you can click on them to fold or expand regions within the document. If there are nested folds present, Option-clicking on the outermost fold will expand or collapse that fold and all subordinate folds.



You can also employ the commands on the View menu to expand or collapse folds, or fold manually-selected ranges of text. (See “The View Menu” on page 75.)

The linear bars displayed in the gutter are range end indicators. They show where each foldable range ends.

When you fold a range, TextWrangler displays a fold indicator within the document to represent that range. To expand the range, you can either click on the disclosure triangle or double-click the fold indicator.



If you expand a range by double-clicking the fold indicator, the entire range will remain selected after expansion.

The View Menu

This menu contains commands which you can use to toggle the display of navigational elements in text windows, to fold and expand regions of text, to select documents, and to get information on documents and files.

Text Display

This submenu contains commands which control various text formatting and display options. You can set key equivalents for any of these commands under the Text Display entry of the View menu in the Menus & Shortcuts preference panel. You can also adjust many of the same options via the Text Options command in the Edit menu.

Show/Hide Fonts

This command toggles display of the standard system font panel, which you can use to set the font, font size, text style, and tab spacing for the active document.

IMPORTANT The chosen display style will be used for *all* text in the window; TextWrangler does not support the use of selective text styles.

Note The font changes you make by using this command affect only the active document. To set the default font, size, style, and tab width for all documents, use the "Default Font" option in the Editor Defaults preference panel.

Soft Wrap Text

This command toggles the use of soft wrapping in the current document. (See "Soft Wrapping" on page 89.)

Show/Hide Page Guide

This command toggles display of the page guide in the current document. (See "Page guide" on page 87.)

Show/Hide Tab Stops

This command toggles display of tab stops in the current document. (See "Tab stops" on page 87.)

Show/Hide Line Numbers

This command toggles display of line numbers in the current document. (See "Line numbers" on page 87.)

Show/Hide Gutter

This command toggles display of the gutter in the current document. (See "Gutter" on page 87.)

Show/Hide Invisibles

This command toggles display of invisible characters in the current document. (See "Show invisibles" on page 87.)

Show/Hide Spaces

This command toggles display of invisible characters in the current document. (See "Show invisibles" on page 87.)

Show/Hide Toolbar

Choose this command to hide or show the toolbar in the frontmost text window. (See “The Toolbar” on page 63.)

Show/Hide Navigation Bar

Choose this command to hide or show the navigation bar in the frontmost text window. (See “The Navigation Bar” on page 65.)

Show/Hide Editor

Choose this command to hide or show the editing pane within a disk browser or results browser window.

Show/Hide Files

Choose this command to hide or show the file list within the frontmost text window. (See “The File List” on page 69.)

Show/Hide Currently Open Documents

You may ignore this command, which is present but always disabled.

Show/Hide Recent Documents

Choose this command to hide or show the “Recent Documents” section within a window’s file list.

Balance

This command locates the pair of parentheses, braces, brackets, or smart (curly) quotes that surround the insertion point or the current selection. If there are unmatched delimiters within this area, TextWrangler beeps. You can also double-click a delimiter character to invoke this command.

When syntax coloring is active for a document, Balance (including auto-balance) will ignore balance characters that appear inside strings or comments.

Balance & Fold

This command behaves identically to Balance (above) except that in addition to locating the paired delimiters, TextWrangler will also generate a fold range including the delimiters and all the text they contain.

Fold Selection

This command generates a fold range from the currently selected text. You can use Unfold Selection (below) or double-click the fold indicator to expand the fold. When there is no active selection, this command is disabled.

Unfold Selection

This command will expand any text folds in the selection range. When there is no active selection, this command is disabled.

Collapse Enclosing Fold

This command will collapse the auto-generated fold that most closely surrounds the current insertion point (or the start of the selection range).

Collapse All Folds

This command will collapse all automatic fold points in the current document.

Expand All Folds

This command will expand all text folds in the current document, whether automatically generated or manually created.

Previous Document/Next Document

You may use these commands to switch between documents within the frontmost text window. (By default, TextWrangler selects documents in most-recently viewed order, but you can choose to have it select documents in name order via an expert preference. For details, see the “Expert Preferences” page in TextWrangler’s built-in Help.)

Move to New Window

Choose this command to open the active document of the frontmost text window into its own text window. If the frontmost text window contains only one document, this command will be disabled.

Open in Additional Window

Choose this command to open the active document of the frontmost text window into an additional text window, while leaving it open in the current window.

Reveal in Finder

Choose this command to open a Finder window which will display the active document’s file. If the active document is not associated with a file, this command will be disabled. Using this command is the same as clicking (without dragging) the document proxy icon in the toolbar.

If the selected text in a document is the name of a file, hold down the Option key as you open the File menu and choose the Reveal Selection command to have TextWrangler open a Finder window which will display that file.

Go Here in Terminal

This command is enabled when the active document has a corresponding disk file. Choose this command to open a Terminal window with the current working directory set to the document's parent directory.

Go Here in Disk Browser

This command is enabled when the active document has a corresponding disk file. Choose this command to open a disk browser in the document's parent directory.

Cursor Movement and Text Selection

TextWrangler gives you several ways to move the insertion point and change the selection. You can click and drag using normal Macintosh text selection techniques or you can use various keys on the keyboard.

Clicking and Dragging

You can select text in an editing window in the normal Macintosh fashion, by clicking and dragging. Holding down the Shift key while clicking or dragging extends the selection.

	No Modifier	Shift
Click	Move insertion point	Extend selection
Double-click	Select word	Extend selection to word
Triple-click	Select line	–none–

Triple-clicking is the same as clicking in a line and then choosing the Select Line command from the Edit menu.

You can hold down the Command or Option keys when clicking or double-clicking to trigger special actions:

	Option	Command
Click	–none–	Open URL
Double-click	Look up selected word in programming reference	–none–

TextWrangler optionally allows you to select entire lines by clicking in the left margin of an editing window. (If you have line numbers displayed, via the Show Line Numbers option in the Appearance preference panel, you can click in the line number as well.) You can click and drag to select multiple lines, double-click to select an entire paragraph, or double-click and drag to select a range of paragraphs. A checkbox in the Editing preference panel, labeled Allow Single-Click Line Selection, controls this behavior. If the checkbox is turned off, clicking in the left margin simply moves the insertion point to the beginning of the clicked line.

Arrow Keys

You can use the arrow keys to move the insertion point right, left, up, and down, and augment these movements with the Command, Option, and Control keys:

	No Modifier	Option	Command	Control
Up	Up one line	Up one screen	Start of document	(moves selected line up)
Down	Down one line	Down one screen	End of document	(moves selected line down)
Left	Left one character	Left one word	Start of line	Previous case transition or word boundary
Right	Right one character	Right one word	End of line	Next case transition or word boundary

Holding down the Shift key extends the selection. For example, pressing Shift-Option-Right Arrow selects the word to the right of the insertion point.

CamelCase Navigation

TextWrangler supports CamelCase navigation. CamelCase (also “camel case”) is the practice of writing intercapitalized compound words or phrases; it is used as a standard naming convention in several programming languages, and as an automatic link creation method in wiki content.

You can move from one part of a CamelCase word to the next by holding the Control key down and tapping the right (or left) arrow key to jump to the next (or previous) transition from lower-case to upper-case or the next word boundary, whichever comes first.

Rectangular Selections

By holding down the Option key as you drag, or holding down the Shift and Option keys while clicking, you can select all text lying within a specified rectangular area (column). You can then perform all of the normal editing operations on this “rectangular selection,” such as Cut, Copy, Paste, or drag and drop, as well as text transformations such as Change Case, Shift Left, Shift Right, Entab, Detab, Increase Quote Level, Decrease Quote Level, Strip Quotes, and Zap Gremlins.

New TextWrangler 5 offers two new commands in the Edit menu: Select Up and Select Down. These commands facilitate rectangular selection via the keyboard. (Their default key equivalents are Control-Shift-up arrow and Control-Shift-down arrow, which can be changed as usual in the Menus & Shortcuts preferences panel.) Starting from either an existing selection that does not cross a line boundary, or an existing rectangular selection, the Select Up command will extend the selection range up, or Select Down will extend it down, within the same column, thus creating (or extending) a rectangular selection.

IMPORTANT Rectangular selection and soft wrapping are mutually incompatible. When soft wrapping is enabled, dragging the mouse to make a selection will always result in a normal (non-rectangular) selection even if you hold down the Option key. Conversely, if you have made a rectangular selection in a hard wrapped document, the Soft Wrap Text option in the Text Options popup menu or sheet will be disabled.

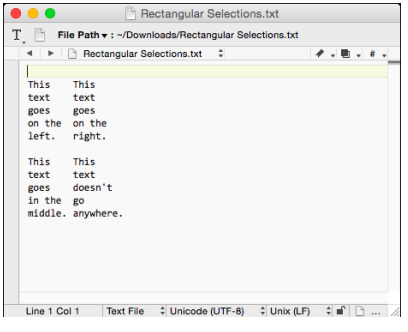
Working with Rectangular Selections

Commonly, while working with text, you will be performing actions on a line-by-line basis; for example, when making a selection, you will start by selecting the contents of one line before moving on to the next. However, if you need to deal with tabular data, it can be useful to think in terms of rectangles or blocks of text that include parts of several lines. This is where you can make use of TextWrangler's ability to manipulate rectangular selections.

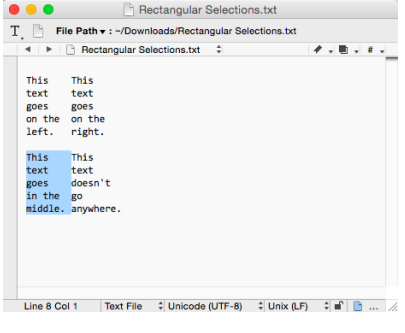
IMPORTANT You cannot make or insert rectangular selections into a document which is soft wrapped, so you must turn off soft wrap before using this technique. (See "Soft Wrapping" on page 89.)

Example: Moving a Column

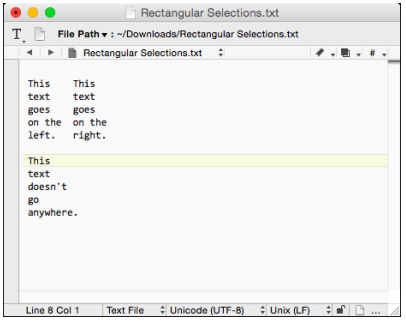
Consider you have the document shown below, and you want to move only the bottom left column (the one that says "This text goes in the middle") and move it in between the top left and top right columns. To do this using standard selection methods, you would have to perform five separate cut-and-paste operations. However, by using rectangular selections, you can move the whole column in one operation.


Rectangular Selections.txt
File Path : ~/Downloads/Rectangular Selections.txt
Rectangular Selections.txt
This text goes on the left. This text goes on the right.
This text goes in the middle. This text goes doesn't go anywhere.
Line 1 Col 1
Text File
Unicode (UTF-8)
Unix (LF)

To start, hold down the Option key while dragging over the bottom left column, until you get a selection that looks like this:


Rectangular Selections.txt
File Path : ~/Downloads/Rectangular Selections.txt
Rectangular Selections.txt
This text goes on the left. This text goes on the right.
This text goes in the middle. This text goes doesn't go anywhere.
Line 8 Col 1
Text File
Unicode (UTF-8)
Unix (LF)

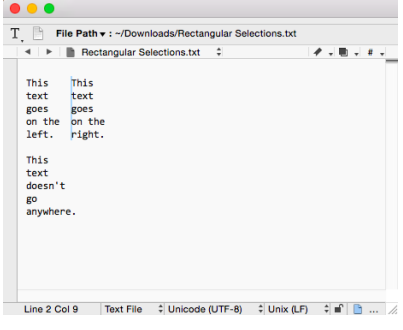
Choose Cut from the Edit menu (or press Cmd-X) to cut the selected text out of the document and place it on the Clipboard.



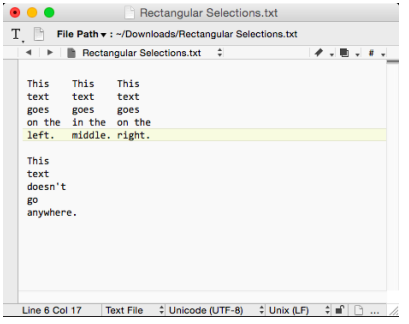
Next, you must paste in the text you just cut. You can do this in either of two ways:

- Use the Paste Column command, which will “paste down” from the current insertion point. This allows you to directly insert text without needing to make a rectangular selection first.
- Make a rectangular selection as described below, and then use the standard Paste command. This procedure is less efficient for moving columnar data than using the Paste Column command, but it allows you to select and replace a region of text as well as simply inserting text.

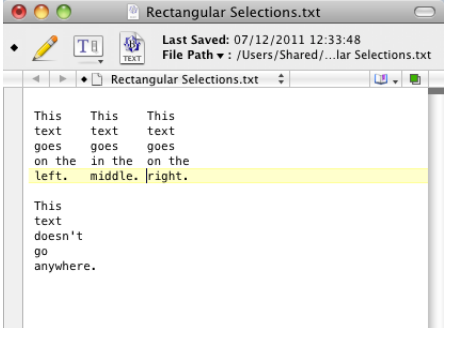
To manually make a rectangular selection prior to pasting text, position the arrow pointer just to the left of the top right column, press and hold the Option key, press the mouse button, and drag straight down until you have a very thin vertical selection just to the left of the whole column, as shown below.



Now, paste the text you previously cut back in, and the task is finished.



Now, paste the text you previously cut back in, and the task is finished.


Rectangular Selections.txt
Last Saved: 07/12/2011 12:33:48
File Path : /Users/Shared/...lar Selections.txt
Rectangular Selections.txt
This text goes on the left.
This text goes in the middle.
This text goes on the right.
This text doesn't go anywhere.

Filling Down

When you apply the Paste Column command and the pasted text contains no line break (or only a single line break at the end), TextWrangler will perform a “fill down”, placing a copy of the pasted text on each line within the selected column.

Further Details

Some word processors also provide support for rectangular selections which works a little differently than TextWrangler's, so you may wish to keep this difference in mind. Typically, when you copy a rectangular selection of text to the clipboard in these programs, they handle that piece of text differently than text copied from a line-by-line selection. Then, when you paste, the text will be entered in a block, even when you have not made a rectangular selection to paste into.

TextWrangler does not do this. Instead, when you copy a rectangular selection to the clipboard, TextWrangler turns the selection into a series of individual lines, which is why you must make a rectangular selection before pasting, so TextWrangler will know it should paste the text in block fashion. Though this method does require an extra step, it is more flexible, because you can select a set of lines and then paste it as a block, or vice versa.

Scrolling the View

Accelerated Scrolling

When clicking the arrows in a scroll bar, you can use the Command and Option keys to accelerate the scrolling. These shortcuts also apply if you use a mouse with a built-in scroll wheel.

Modifier	Scroll Speed
none	Normal
Command	2x accelerated
Option	3x accelerated
Command+Option	6x accelerated

The Delete Key

The Delete key deletes the character to the left of the insertion point. If you have selected text, the Delete key deletes all the text in the selection. You can use the Command and Option keys to modify the way the Delete key works:

Modifier	Action
none	Deletes character to the left of the insertion point
Option	Deletes to the beginning of the word to the left of the insertion point
Command	Deletes to the beginning of the line
Command+Option	Deletes to the beginning of the document

Holding down the Shift key with the Delete key makes the Delete key work the same way as the Forward Delete key on extended keyboards.

The Numeric Keypad

Some keyboards have a numeric keypad on the right side. Normally, you use the keys on the keypad to enter numbers.

To toggle the behavior of the keypad between moving the cursor and entering numbers, hold down the Option key and press the Clear key in the upper-left corner of the keypad. (This key is also labeled Num Lock on some keyboards.)

When keypad navigation is active, TextWrangler will perform the following actions:

start of line **7**	up **8**	Scroll up **9**
left **4**	show selection **5**	Right **6**
end of line **1**	down **2**	Scroll down **3**

You can use the Shift key with the keys on the numeric keypad to extend a selection. You can use the Command and Option keys with the 2, 4, 6, and 8 keys as you would the arrow keys.

Line Number Command

To move the insertion point to a specific line, use the Line Number command in the Go menu. When you choose this command, TextWrangler opens a sheet in the frontmost document. Type the number of the line you want to move to and click Go To.

The Line Number command will also accept relative inputs. Entering a value prefixed with +/- will add that value to the current line number. For example, with the insertion point in line 100, "+75" will move to line 175; "-75" will go to line 25. (As always, when you enter an unsigned number, TextWrangler will move to the specified line number.)

Note The Line Number command honors the Use "Hard" Lines in Soft-Wrapped Views option in the Editing preference panel.

Function Keys

If your keyboard has function keys, you can use the following key equivalents for cutting and pasting, to scroll, and to move the insertion point.

	No Modifier	Option	Command	Shift
del	forward delete	delete to end of word	delete to end of line	
Home	scroll to top of document		move insertion point to start of document	
End	scroll to end of document		move insertion point to end of document	

	No Modifier	Option	Command	Shift
Pg Up	scroll page up			
Pg Dn	scroll page down			

Note Holding down the Command and Option keys as you press the forward delete key deletes to the end of the document.

Resolving URLs

To resolve a URL (Uniform Resource Locator), you can Command-click anywhere in the URL text, or Control-click to bring up the contextual menu and choose Open URL from the menu. TextWrangler will examine the URL and launch the appropriate helper application. If the URL is not valid or the helper application cannot be found, TextWrangler will beep.

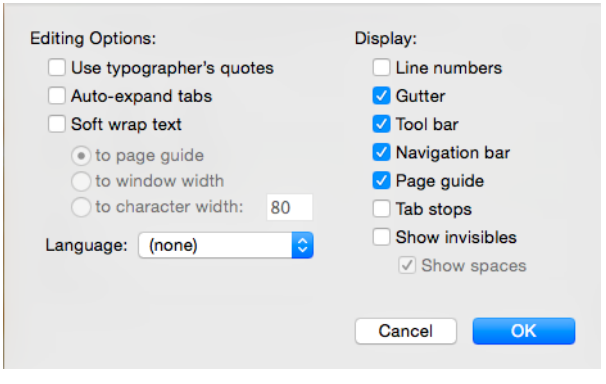
Note Some Web browsers cannot resolve URLs if the request is sent when the browser is starting up. If your Web browser does not properly resolve the URL, wait until the browser has finished starting up and then try again.

Bare Bones Software gratefully acknowledges John Norstad for providing the URL parsing code.

Text Options

You can use the Text Options command in the Edit menu to change the way TextWrangler edits text and the way it displays text and additional elements in its windows. When you choose this command, TextWrangler will drop a Text Options sheet in the current text window.

The controls on the Text Options sheet are divided into two parts: the Editing options on the left control the way TextWrangler behaves while you type, and the Display options on the right control the appearance of the TextWrangler window.



Note You can also change many of these options using the commands in the Text Display submenu of the View menu.

Changes you make in the Text Options sheet affect only the active document or window. To set options which will apply to all text windows you open, use the Editor Defaults and Appearance preference panels.

Editing Options

These options control the way TextWrangler behaves as you type text in the active document window. Changes you make here affect only that document. To change the default editing options for documents that you will open in the future, use the Editor Defaults preference panel.

Use typographer's quotes

When this option is on, TextWrangler will automatically replaces straight quotes (" ') with typographer's quotes (“ ” ‘ ’) in the current document. If you need to type a straight quote when this option is selected (or to type a typographer's quote when the option is not selected), hold down the Control key as you type the " or ' key.

Note We recommend against using this option if you are editing HTML content, email content, or program code.

Auto-expand tabs

When this option is selected, TextWrangler inserts an appropriate number of spaces when you press Tab, rather than inserting a tab character.

Additionally, when there are only spaces (and tabs) between the insertion point and the start of the current line (or the first non-whitespace character on the line), TextWrangler will delete a tab stop's worth of spaces when you press Delete (Backspace).

Soft wrap text

When this option is selected, TextWrangler soft-wraps the text in the file to the right margin that you choose: the page guide, the window width, or a specific number of characters. The page guide is an arbitrary visual boundary whose width you can set in the Appearance preference panel. (See “Soft Wrapping” on page 89 to learn how wrapping works in TextWrangler.)

Language

The Language menu lets you specify which source code language the file uses. The file's language setting affects how TextWrangler performs syntax coloring and parses function names for the function popup menu. TextWrangler generally determines the file's language from its filename extension, using the mapping table in the Languages preference panel.

For example, “.cp” files are C++, and “.m” files are Objective-C. You can use this menu to override those settings for a specific file. To quickly check the language for a file, click the Text Options popup menu in the toolbar and look at the Languages item.

Display Options

These options determine which controls appear in the frontmost text window, regardless of whether that window contains one or more documents. Changes you make here affect only that window. To change the display characteristics for text windows that you will open in the future, use the Appearance preference panel.

Line numbers
This option displays line numbers along the left edge of the window.

Gutter
This option shows or hides the gutter in the window.

Toolbar
This option shows or hides the toolbar in the window.

Navigation bar
This option shows or hides the navigation bar in the window.

Page guide
This option shows or hides the page guide in the window.

Tab stops
This option shows or hides tab stop indicators in the window.

Show invisibles
This option shows or hides non-printing characters in the window. Select this option when you want to see line breaks, tabs, and "gremlins" (other invisible characters). TextWrangler uses these symbols:

Symbol	Meaning
∆	tab
◊	space
●	non-breaking space
¬	line break
¶	page break
¿	other non-printing or special characters

If you turn on Show Invisibles, the Show Spaces option will become available, allowing you to enable display of the visually "noisy" space characters if you desire.

Syntax Coloring
When this option is selected and the editing window contains a document in a programming language TextWrangler recognizes, TextWrangler displays keywords and other language elements in color.

TextWrangler uses several methods to determine what language (if any) to use for a particular file. The primary way to activate syntax coloring in a document is simply to save it with a file name extension that indicates what programming or markup language the file contains. For example, if you save your file with ".html" at the end of the file name, TextWrangler will color your HTML tags and anchors. Some other common suffixes are ".text" for Markdown files, ".py" for Python files, and ".rb" for Ruby files.

For any file whose name does not have an extension, or whose name has an extension that does not match any of the mappings in TextWrangler's Languages preference panel, TextWrangler will attempt to guess what language the file contains and apply the appropriate syntax coloring. If TextWrangler guesses wrong (or is unable to guess), you can resort to the Language submenu of the Text Options popup menu in the toolbar or the Language popup menu in the Text Options sheet, which gives you the ability to manually select *any* installed language to be applied to the document, regardless of its name. If you then save the file, your manual language selection will persist and override any suffix mapping.

By default, TextWrangler recognizes over 20 different languages and several dozen suffix mappings. You can add new suffixes to map to existing languages or (by installing third-party language plug-ins) add syntax coloring support for new languages as well. All the specific languages that TextWrangler recognizes, and the suffixes or extensions it expects for them, are listed in the Languages preference panel, and suffix mappings can also be changed there. You can choose the colors that TextWrangler uses for syntax coloring in the Text Colors preference panel.

Note TextWrangler will recognize and syntax-color VBScript embedded within HTML via the <%…%> and <SCRIPT>…</SCRIPT> tags.

How TextWrangler Wraps Text

TextWrangler wraps text in one of two ways: soft wrapping or hard wrapping.

Soft wrapping is like the word wrapping found in most word processors. When the insertion point reaches a right margin as you type, the word processor automatically moves the insertion point to the beginning of the next line. You never need to type a carriage return (that is, press the Return key) at the end of a line, but only to start a new paragraph. If you place the insertion point in the middle of a paragraph and start typing, the text reflows so that words that are pushed out beyond the right margin end up on the next line. Usually, you use soft wrapping when you are editing memos, mail messages, and other prose. It is also useful for HTML documents. With soft wrapping, you generally do not have to scroll the window horizontally to see all the text in the file.

Unlike soft wrapping, hard wrapping requires a carriage return at the end of every line. When soft wrapping is turned off, TextWrangler lets you type as far as you like on a line, and never automatically moves the insertion point to the beginning of the next line. You have to manually type a carriage return to start a new line. You usually use hard wrapping to write programs, tabular data, resource descriptions, and so on. With hard wrapping, each line of source code or data appears on its own line in the window, although you may have to scroll the window horizontally to see the entire line if it is long.

Note When you use the Hard Wrap command on a rectangular selection, lines will be padded with spaces as necessary.

Tip If you open a file in TextWrangler that appears to consist of a few very long lines, you should select the soft wrapping option for that file.

This table summarizes the commands to soft-wrap and hard-wrap text. The sections that follow give details about using the wrapping commands.

To do this…	Do this…
Soft-wrap text as you type	Choose Soft Wrap Text from the Text Display submenu of the View menu or select the Soft Wrap Text option from the Text Options sheet
Convert hard-wrapped text to soft-wrapped text	Use the Remove Line Breaks command in the Text menu, and activate soft wrapping
Convert soft-wrapped text to hard-wrapped text	Use the Add Line Breaks command in the Text menu
Hard-wrap text to a specific margin, reflowing paragraphs as needed	Use the Hard Wrap command in the Text menu

Soft Wrapping

To turn on soft wrapping for the active window do one of the following:

- Choose Soft Wrap Text from the Text Display submenu of the View menu.
- Select the Soft Wrap Text option from the Text Options sheet by choosing Text Options from the Edit menu.

To specify the wrapping margin, use the Text Options command. You can have text wrap at the Page Guide, the edge of the window, or a specific character position.

IMPORTANT Soft wrapping and rectangular selection are mutually incompatible. When soft wrapping is enabled, dragging the mouse performs normal (non-rectangular) selection even if the Option key is held down; when there is a rectangular selection, the Soft Wrap Text option is unavailable in the Text Options popup menu and dialog box.

To make soft wrapping the default for new windows, select the Soft Wrap Text option in the Editor Defaults preference panel. You can also use the settings in that panel to specify the default wrapping margin.

To “freeze” the current line endings and hard-wrap the text at the current soft wrapping settings, use the Add Line Breaks command to insert a carriage return at the end of each line.

While TextWrangler prefers to break lines at white space when soft-wrapping, lines will be broken as close as possible to the designated wrap width if they do not contain any white space. This way, long URLs and other extended strings of characters are visible without requiring horizontal scrolling.

Soft Wrapping with Indentation

You can control how TextWrangler indents soft wrapped text by means of the Soft Wrapped Line Indentation option in the Editing preference panel. Choose Flush Left to have all lines of each paragraph below the first wrap flush to the left margin of the window. Choose First Line to have all subsequent lines of a paragraph wrap to the same indent level as its first line. Choose Reverse to have all subsequent lines of each paragraph wrap indented one level deeper than its first line.

Exporting Soft-Wrapped Text

TextWrangler will not insert hard line breaks into softwrapped files upon saving them. If you wish to add hard line breaks to a softwrapped file, use the Hard Wrap or Add Line Breaks command.

Soft Wrapping in Browsers

Use the Text Options command from the Edit menu to control soft wrapping (and other display options) for files viewed in a browser window.

Soft Wrapping and Line Numbers

The preference Use “Hard” Lines in Soft-Wrapped Views controls how line numbers are displayed when you use soft wrapping. If this option is turned on, the line number bar, cursor position display, and Line Number commands in editing views will use line numbers that correspond to “hard” carriage returns in the document, rather than to soft-wrapped line breaks. To restore the behavior of previous versions of TextWrangler, turn this preference off.

Hard Wrapping

The easiest way to hard-wrap text is to type a carriage return (by pressing the Return key) whenever you want to start a new line. If you are editing program source code, it is generally best to turn off soft wrapping altogether.

To turn off soft wrapping for the active window, do one of the following:

- Choose Soft Wrap Text from the Text Options popup menu in the toolbar.
- Deselect the Soft Wrap Text option from the Text Options sheet box by choosing Text Options from the Edit menu.

To turn off soft wrapping for new windows, deselect the Soft Wrap Text option in the Editor Defaults preference panel.

TextWrangler provides two ways to convert soft-wrapped text into hard-wrapped text. The first is a simple technique that uses a single command; the second is a bit more complicated but gives you much more control over wrapping.

Hard-Wrapping Soft-Wrapped Text

To convert soft-wrapped text to hard-wrapped text, use the Add Line Breaks command in the Text menu. This command inserts a carriage return at the end of every line of the text as it appears in the window. If your wrapping margin is the edge of the window, you will get different results depending on the width of the window.

If the current document contains a selection range, Add Line Breaks will affect only the selected text; if there is no selection, this command will affect the entire contents of the current document.

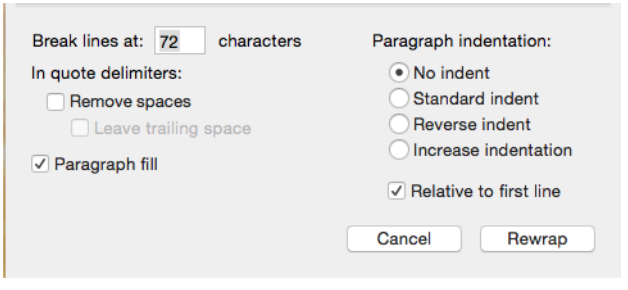
Note The Add Line Breaks command does not turn off soft wrapping.

Hard Wrapping and Filling Text

The Hard Wrap command in the Text menu offers more flexibility for hard-wrapping text than the Add Line Breaks command. Whereas Add Line Breaks merely "freezes" the line breaks displayed in a document by inserting carriage returns, the Hard Wrap command allows you to wrap text to any arbitrary width, while also reflowing or indenting paragraphs.

If the current document contains a selection range, Hard Wrap will affect only the selected text; if there is no selection, this command will affect the entire contents of the current document.

When you choose the Hard Wrap command, TextWrangler opens a sheet in the frontmost document:



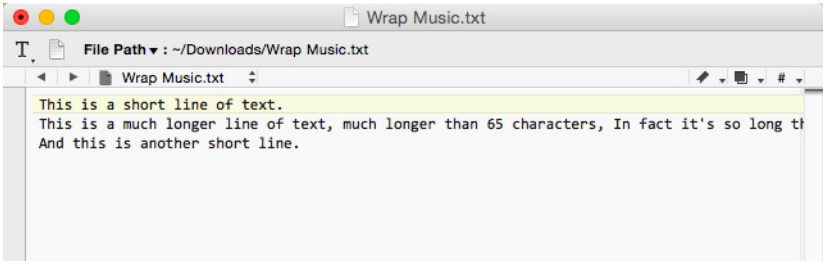
The controls in the left half of the sheet control the maximum width of lines after hard wrapping, and whether wrapped lines should be consolidated to fill paragraphs to the specified width. The controls in the right half determine how paragraphs should be indented.

The "Break Lines at" buttons let you specify the wrapping margin.

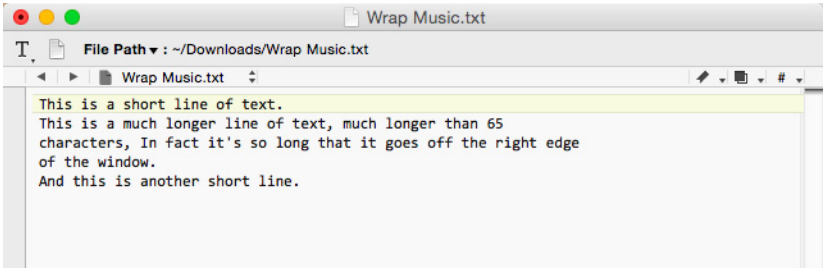
If the text contains Internet-style quotes (one or more ">" characters at the beginning of each line) and the "Remove spaces" option is selected, TextWrangler will remove the leading and trailing spaces from each line, or if the "Leave trailing space" option is also selected, TextWrangler will remove the leading spaces while leaving trailing spaces.

If the Paragraph Fill option is selected, TextWrangler forms the lines into paragraphs before wrapping the lines. An example is the best way to illustrate this option.

Suppose you start with this text:

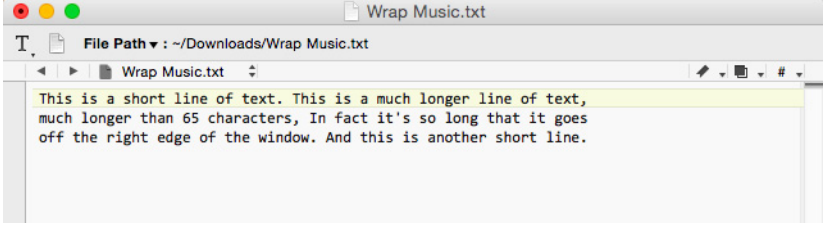

Wrap Music.txt
File Path : ~/Downloads/Wrap Music.txt
Wrap Music.txt
This is a short line of text.
This is a much longer line of text, much longer than 65 characters, In fact it's so long th
And this is another short line.

This is what happens when you wrap to 65 characters with Paragraph Fill off:


Wrap Music.txt
File Path : ~/Downloads/Wrap Music.txt
Wrap Music.txt
This is a short line of text.
This is a much longer line of text, much longer than 65
characters, In fact it's so long that it goes off the right edge
of the window.
And this is another short line.

TextWrangler breaks the long line at a width of 65 characters (twice, because the line was so long) and leaves the short lines alone.

This is what happens to the same text when you wrap with Paragraph Fill on:


Wrap Music.txt
File Path : ~/Downloads/Wrap Music.txt
Wrap Music.txt
This is a short line of text. This is a much longer line of text,
much longer than 65 characters, In fact it's so long that it goes
off the right edge of the window. And this is another short line.

TextWrangler joins all the lines together to form a single paragraph and then wraps the text to a width of 65 characters.

The Paragraph Indentation buttons let you indent paragraphs after they have been wrapped.

- Flush Left does not indent paragraphs at all.
- First Line indents all lines in the paragraph by one tab stop.
- Reverse places the first line in the paragraph flush against the left edge of the window and indents all other lines in the paragraph by one tab stop.

Mark the Relative to First Line checkbox to make any paragraph indents relative to the original indent of the first line of the selection or document. If you want paragraph indents to be relative to the left margin of the document, make sure this checkbox is not marked.

Click the Wrap button to perform the Hard Wrap command, or cancel to dismiss the sheet.

Tip If you hold down the Option key as you choose the Hard Wrap command, TextWrangler uses the last Hard Wrap settings to perform the operation, without displaying a sheet.

The Insert Submenu

In addition to typing, you can use the commands in the Insert submenu of the Edit menu to insert text into the active window. These commands, which are also available in the Insert popup menu (left) in the document toolbar, let you insert the contents of other files, folder listings, Macintosh Toolbox templates, and page break characters.

Inserting File Contents

The File Contents command inserts the contents of one or more files into the document you are editing. When you use this command, TextWrangler displays an Open sheet in which you can choose the files to insert. To select more than one file hold down the Shift key or Control key as you click the files. TextWrangler then inserts the contents of the selected files at the insertion point or replaces the selected text. If you select more than one file, the files will be inserted in alphabetical order, according to file name.

Tip You can also drag a file's icon from the Finder into a TextWrangler editing window to insert the contents of that file.

Inserting File & Folder Paths

The File/Folder Paths command inserts the full path information for the selected files and folder into the document you are editing. When you use these commands, TextWrangler displays a sheet that lets you select the files and/or folders. TextWrangler inserts the path information at the insertion point or replaces the selected text.

Inserting a Folder Listing

The Folder Listing command inserts a textual listing of a folder hierarchy. When you use this command, TextWrangler displays a sheet that lets you select a folder to insert, and inserts that folder's listingat the insertion point or replaces the selected text.

Tip You can also drag a folder's icon from the Finder into a document to insert a folder listing.

Inserting a Page Break

To insert a page break, choose the Page Break command from the Insert submenu of the Edit menu. This will place a form feed character (ASCII 12) at the location of the insertion point. TextWrangler uses this character to indicate the start of a new page when printing.

Inserting Time Stamps

To insert the current time, choose Short Time Stamp or Full Time Stamp from the Insert submenu of the Edit menu. These commands will insert short and long forms (respectively) of the current date and time at the location of the insertion point.

Inserting an Emacs Variable Block

To insert an Emacs variable block describing the option settings for the current document, choose Emacs Variable Block from the Insert submenu of the Edit menu. This will bring up a sheet which you can use to review and confirm the desired options. (Since depending on what options are set, the resulting block can be rather verbose, you may wish to prune the resulting text.)

These options specified in this block will take precedence over saved document state when TextWrangler opens the document. (Inserting these explicit settings can be useful when sharing the document with others.)

Comparing Text Files

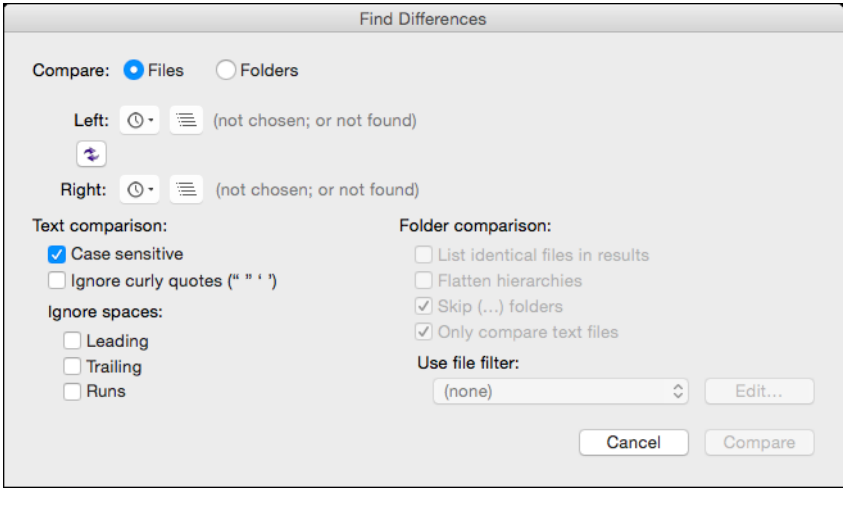
If you have ever had to reconcile changes between two different versions of a file, or even larger numbers of documents, you know how laborious this task can be. TextWrangler's Find Differences command is a powerful tool for doing such comparisons faster and more effectively. Using Find Differences, you can compare any two files, or the contents of two folders. You can also specify options to eliminate minor variations in document content, such as different amounts of white space, from being considered.

If you have two or more text documents open, choose the Compare Two Front Windows command on the Search menu to quickly compare the topmost two documents. (TextWrangler will automatically determine which document is newer and which older based on their modification dates.)

To compare two arbitrary files or folders:

1 **Choose the Find Differences command from the Search menu.**

TextWrangler opens the Find Differences dialog box.



2 **Click the Compare Files radio button.**

3 **Use the Left and Right popup menus to select the documents you want to compare.**

If the files you want to compare are already open, they will appear in the popup menus; otherwise, you can select them by clicking the standard item selection button next to either popup menu, or by dragging the icon of any desired file or folder icons from the Finder into the Left or Right field.

You can also select recently opened files from the Recent Files item on the New and Old popup menus.

4 **Select the Compare options that apply.**

When the Case Sensitive option is selected, TextWrangler distinguishes uppercase from lowercase letters; deselect this option if you want TextWrangler to consider uppercase and lowercase letters the same.

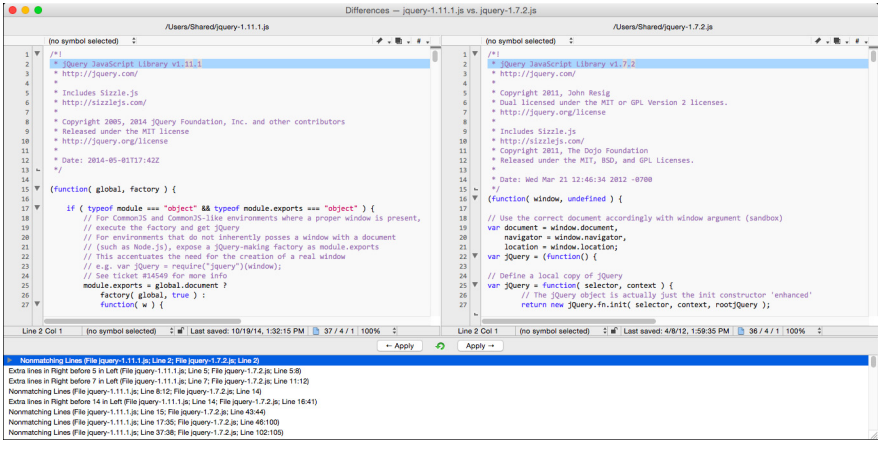
When Ignore Curly Quotes is selected, TextWrangler treats typographers' quotes the same as straight quotes.

When one or more of the Ignore Spaces options is selected, TextWrangler will ignore the corresponding presence of whitespace at the specified positions while comparing files.

5 **Click Compare to perform the comparison.**

Alternatively, you can use the 'bbdiff" command line tool to specify two files or folders, and have TextWrangler perform a Find Differences on them.

If the two files are different, TextWrangler tiles the documents and opens a Differences window below them.


Differences — jquery-1.11.1.js vs. jquery-1.7.2.js
/Users/Shared/jquery-1.11.1.js
/Users/Shared/jquery-1.7.2.js
Nonmatching Lines (File jquery-1.11.1.js; Line 2; File jquery-1.7.2.js; Line 2)
Extra lines in Right before 5 in Left (File jquery-1.11.1.js; Line 5; File jquery-1.7.2.js; Line 5:8)
Extra lines in Right before 7 in Left (File jquery-1.11.1.js; Line 7; File jquery-1.7.2.js; Line 11:12)
Nonmatching Lines (File jquery-1.11.1.js; Line 8:12; File jquery-1.7.2.js; Line 14)
Extra lines in Right before 14 in Left (File jquery-1.11.1.js; Line 14; File jquery-1.7.2.js; Line 16:41)
Nonmatching Lines (File jquery-1.11.1.js; Line 15; File jquery-1.7.2.js; Line 43:44)
Nonmatching Lines (File jquery-1.11.1.js; Line 17:35; File jquery-1.7.2.js; Line 46:100)
Nonmatching Lines (File jquery-1.11.1.js; Line 37:38; File jquery-1.7.2.js; Line 102:105)

The Differences window lists all the differences between the new file and the old file. To see the differences in context, click a line in the Differences window; TextWrangler scrolls and selects that spot in both files.

The range of lines within each file which belong to the selected difference are highlighted with the standard selection color, while individual differences within each difference range are drawn with a grey background.

Reviewing and Applying Differences

To view and apply individual differences within a line or region (i.e. sub-line differences), just click on the triangle to the left of that difference to expand the list and select the appropriate character difference.

Use the Apply to Left and Apply to Right buttons in the Differences window to transfer the differing text from the new file to the old file, or vice versa. After you use one of these buttons, TextWrangler italicizes the entry in the Differences window to indicate that you have already applied that change.

You may also apply all differences by clicking in the differences list, then choosing Select All in the Edit menu, and using the Apply to Left or Apply to Right button to apply the differences to the desired file.

If a Differences window is open and is the frontmost window, the Compare Again command in the Search menu will recompare the two files being compared and refresh the list of differences accordingly. The small button (with the circular icon) between the Apply to Left and Apply to Right buttons performs the same function.

Preserving a List of Differences

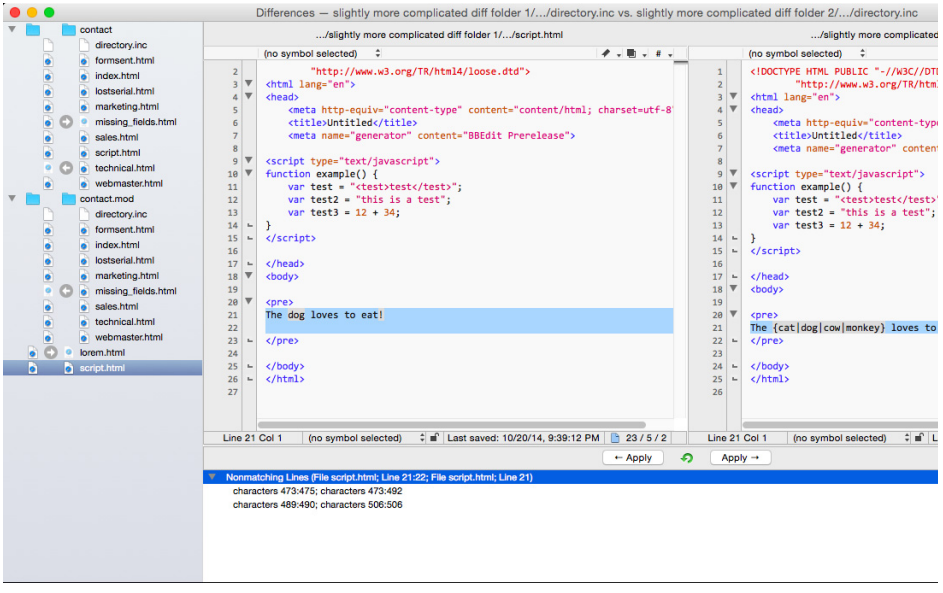
When a Differences window is active, you can save the currently listed differences to a plain text file by using the Export command in the File menu.

Compare Against Disk File

You can use the Compare Against Disk File command to compare the contents of the active document against the disk file for that same document. This capability makes it easy to locate in-progress changes to a document.

Multi-File Compare Options

You can compare multiple files at once by selecting the Folders button in the Find Differences dialog; TextWrangler lists all the files in each folder, and displays a directional arrow icon to denote files which exist only in one folder.



You can select any file pair to view their differences (as for a single pair of files), or for files which exist only in one folder, click the arrow icon to copy the existing file into the corresponding location in the second folder. In addition, you can click any file's icon to ask the OS to open it, or Option-click to reveal that file in the Finder, or Command-click to open that file into a separate editing window within TextWrangler.

When performing a multiple file comparison, you can specify the additional options described below.

List identical files

Normally, when you compare folders using the Find Differences command TextWrangler presents you with three lists: one list of the items that are in the first folder but not in the second folder, another list of the items that are in the second folder but not in the first one, and another list of the items that appear in both folders.

The list of items that appear in both folders generally displays a bullet next to items that are not identical. For example, if you have an archived mail folder that you are comparing against a current mail folder, mailbox files that appear in both the old and new file will all be listed together; however, if there have been any changes to the contents of particular mailbox files, the changed mailbox files will be listed with bullets next to them.

If you are comparing very large folders, however, the list of common items can be extremely long, making the flagged items hard to find. When you deselect the List Identical Files checkbox, TextWrangler will list only the flagged items (the ones that have been changed) in the list of items that appear in both folders.

Flatten hierarchies

Normally, TextWrangler retains the hierarchy of the files being compared in a folder. In other words, when comparing folders, it looks in each subfolder of the first folder you select and tries to match it with a file of the same name in the same subfolder of the second folder, and so on down for all subfolders. If you choose Flatten Hierarchies, TextWrangler considers the files in the folders as a single flat list, allowing a file in one folder to match a file of the same name in the other folder, regardless of whether they are in the same subfolder in both hierarchies.

Skip (...) folders

If this option is set, TextWrangler skips subfolders whose names are enclosed in parentheses when comparing folders.

Only compare text files

If this option is set, TextWrangler does not include non-text files when comparing folders.

Use file filter

File filters allow you to select files for comparison with great precision. If either file in a compared pair matches the filter, the files are eligible for comparison; if *neither* file matches the filter, the files will not be compared. See Chapter 7, "Searching," for more information on creating, editing, and using file filters.

Note When comparing folders with the Find Differences command, TextWrangler applies any specified file filter to the contents of the resulting "Only in new" and "Only in old" lists, so that only those files that match the filter criteria will appear in the lists.

Using Markers

A marker is a selection range that you can name. If a document contains any markers, you can select them from the Mark popup menu to move quickly to the specified section of the file. (The navigation bar must be visible in order to access the Mark popup menu. Choose Show Navigation Bar from the View menu to display the Navigation bar if it's hidden.)

Note If you are programming, you may be tempted to use markers to mark functions in your source code. However, if TextWrangler supports the language you are using, this is usually unnecessary; your functions will automatically appear in the Function popup menu in the document window.

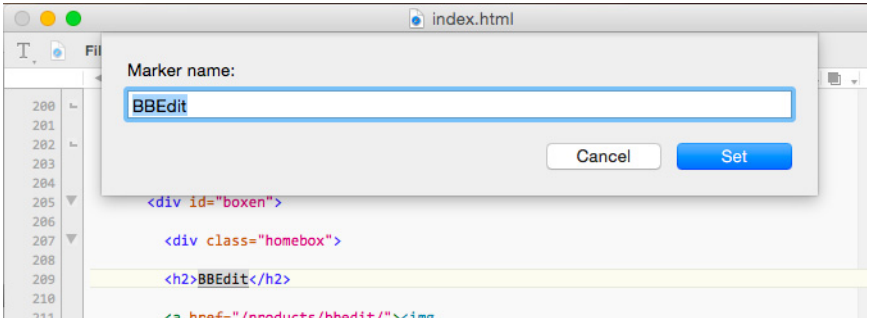
Setting Markers

To set a marker:

1 **Select the text you want to mark.**

2 **Choose the Set Marker command from the Mark popup menu (identified by the icon shown at left), or Control-click the selected text and choose Set Marker from the contextual menu.**

TextWrangler opens a sheet so that you can name the marker. If you have selected a range of text, the sheet will contain the first characters of the selection.



3 **Click Set to set the marker.**

Tip If you hold down the Option key as you choose Set Marker, TextWrangler sets the marker using the leading characters of the selected text as the name of the marker, without displaying a dialog box.

Clearing Markers

To clear a marker:

1 **Choose the Clear Markers command from the Mark popup menu.**

TextWrangler displays the list of markers.

2 **Select the marker you want to delete.**

3 **Click Clear to clear the marker.**

TextWrangler also offers a Clear All Markers command, which clears all the markers in the document in one fell swoop. You can access this command by holding down the Option key and using the Mark popup menu.

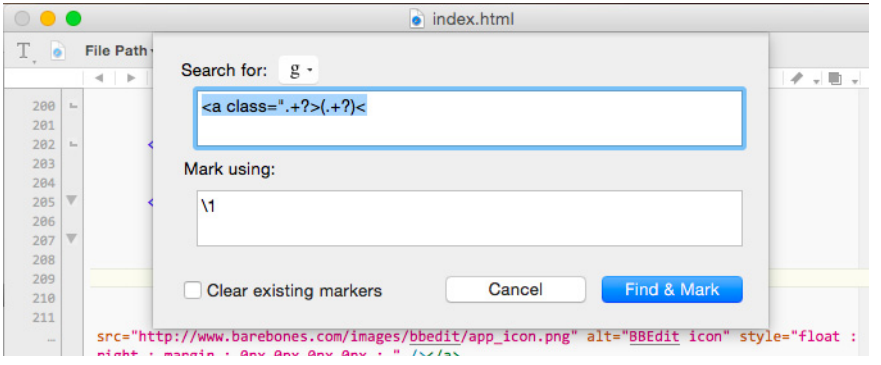
Using Grep to Set Markers

You can use the Find & Mark All command in the Mark popup menu to mark text that matches a grep pattern. To learn more about using grep patterns, see Chapter 8, “Searching with Grep.”

To use a grep pattern to mark text:

1 **Choose the Find & Mark All command from the Mark submenu.**

TextWrangler opens the Find & Mark All sheet.



2 **Type the pattern in the Search For field and the marker names in the Mark With field.**

You can also choose stored patterns from the Patterns popup menu.

3 **Click Find & Mark to mark the matching text.**

TextWrangler searches the current document for text that matches the pattern and marks it the way you specified.

Spell Checking Documents

The Check Spelling command in the Text menu lets you check the spelling of the text in your documents using the system's built-in spelling checker.

Check Spelling As You Type

To have TextWrangler automatically check spelling as you type for the current document, select Check Spelling as You Type in the Text menu. To have TextWrangler always check spelling as you type, turn on the corresponding option in the Editor Defaults preference panel.

When TextWrangler encounters a word which is either misspelled or not in the checker's dictionary, it will draw a heavy red underline beneath the word. You can either type a correction, or Control-click on the word and select a suggested correction from the contextual menu.

To skip the identified word and continue checking, use the Check Spelling command again. To ignore all further instances of the word, Control-click on it and choose Ignore Spelling from the contextual menu. To add the word to the dictionary, Control-click on it and choose Learn Spelling from the contextual menu.

Manual Spell Checking

Choose the Find Next Misspelled Word command from the Text menu, or type its key equivalent (Command-;) to start checking a document's spelling. TextWrangler will check every word in the document in order, starting from the current insertion point.

To check the spelling of all words in the document at once, choose the Find All Misspelled Words command, or type its key equivalent (Command-Option-;). TextWrangler will draw an underline under every questioned word in the document. You can then correct the spelling of any questioned word by typing, or by using the contextual menu to select a suggested correction or to skip, ignore, or add the word to the dictionary.

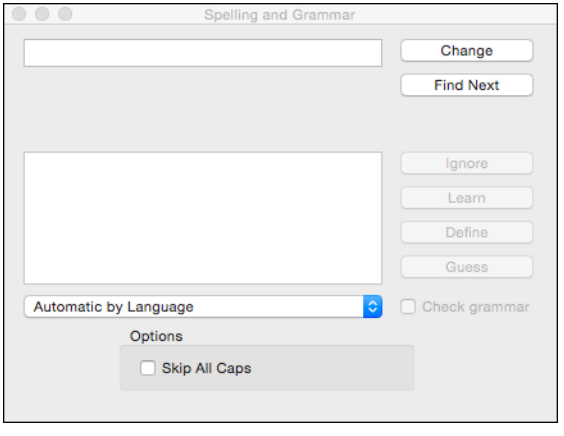
To clear the underline from all questioned words, choose the Clear Spelling Errors command.

The Spelling Panel

In addition to allowing you to correct, ignore, or learn identified words, the Spelling panel allows you to choose which spelling dictionary TextWrangler will use, and to forget learned spellings. To use the Spelling panel:

1 **Choose the Show Spelling Panel command from the Text menu.**

TextWrangler opens the Spelling panel.



2 **Set spelling options.**

Choose a dictionary to use by selecting it from the Dictionary popup menu. Select Skip All Caps to avoid checking words consisting of only capital letters. (Note that these settings persist across runs of the application.)

3 **Click Find Next to begin checking.**

TextWrangler scans the document, and stops at the first misspelled or unrecognized word. This word is displayed in the text field to the left of the Correct button. Possible corrections for the questioned word are listed in the Guess box above.

4 **If the questioned word is misspelled, choose the correct spelling from the Guess list or type it yourself in the Correct field.**

5 Click one of the Spelling panel's action buttons to handle the questioned word.

Click Ignore to ignore further instances of the questioned word, without adding it to the active dictionary.

Click Guess to display a list of possible corrections.

Click Find Next to ignore this instance of the questioned word and continue checking.

Click Correct to replace this instance of the questioned word with the text in the adjacent text field.

Click Learn to add the questioned word to the active dictionary.

Click Forget to remove the questioned word from the active dictionary.

CHAPTER

5 Text Transformations

This chapter describes the range of powerful text transformation commands offered by TextWrangler.

In this chapter

Text Menu Commands **103**

Apply Text Filter – 103 • *Exchange Characters* – 104
Change Case – 104 • *Shift Left / Shift Right* – 105
Un/Comment Selection – 105 • *Hard Wrap* – 105
Add Line Breaks – 106 • *Remove Line Breaks* – 106
Convert to ASCII – 106 • *Educate Quotes* – 106
Straighten Quotes – 106 • *Add/Remove Line Numbers* – 106
Prefix/Suffix Lines – 107 • *Sort Lines* – 107
Process Duplicate Lines – 108 • *Process Lines Containing* – 109
Increase and Decrease Quote Level – 110 • *Strip Quotes* – 110
Zap Gremlins – 110 • *Entab* – 112 • *Detab* – 112
Normalize Line Endings – 112

Text Menu Commands

TextWrangler provides a variety of commands which you can use to transform text in different and useful ways. Most of these commands are situated in the Text menu, and described in this section. You can also use TextWrangler's search and replace capabilities, or additional plug-in tools, to transform text; each of these topics is covered in a separate chapter.

Unless otherwise specified, each of these commands will be applied to the active text selection in the frontmost document range, or if there is no active selection, to the entire contents of the document.

Hold down the Option key when selecting any command from the menu in order to quickly re-invoke it with its last-used option settings. (These "short form" commands are also available in the Menus & Shortcuts preference panel, so that you can set key equivalents for them.)

Apply Text Filter

This command presents a submenu listing all currently available text filters. (These filters consist of any executable items contained in the Text Filters folder of TextWrangler's application support folder. See "Text Filters" on page 31.)

When you choose a filter, TextWrangler will pass either the selected text (or the contents of the active document, if there is no selection) on STDIN to Unix executables or filters or as a reference to a 'RunFromTextWrangler' entry point in AppleScripts. (If an AppleScript script does not have a 'RunFromTextWrangler' entry point, TextWrangler will call its run handler, again passing a reference to the current selection range.)

AppleScript scripts should return a string which TextWrangler will use to replace the selection range, while Unix filters should write to STDOUT.

Note TextWrangler no longer supports text factory execution.

Apply Text Filter <last filter>

This command will reapply the two characters according to the following rules:

Exchange Characters

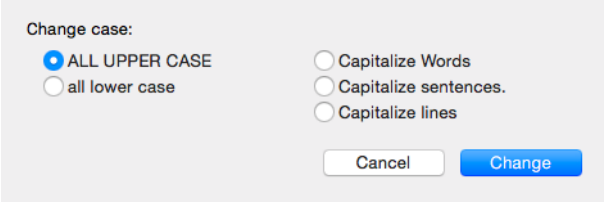
This command swaps two characters according to the following rules:

- If there is no selection and the insertion point is not at the beginning or end of a line or of the document, this command transposes the two characters on either side of the insertion point.
- If the insertion point is at the beginning of a line or document, this command transposes the two characters following the insertion point.
- If the insertion point is at the end of a line or document, this command transposes the two characters before the insertion point.
- If there is a selection, this command transposes the characters at *either end* of the selection.

If you hold down the Option key as you choose this command, Exchange Characters becomes Exchange Words. Exchange Words behaves like Exchange Characters except that it acts on entire words rather than individual characters.

Change Case

This command lets you change between uppercase and lowercase characters, or capitalize word, line, or sentence starts. When you choose the Change Case command, the following sheet appears:



The radio buttons let you choose how to change the case of the text. The following table explains the function of each option in this dialog.The radio buttons let you choose how to

This button…	Changes the text like this…
ALL UPPER CASE	Every character changes to uppercase.
all lower case	Every character changes to lowercase.
Capitalize Words	The first character of every word changes to uppercase; all other characters change to lowercase.
Capitalize sentences	The first character of every sentence changes to uppercase; all other characters change to lowercase.
Capitalize lines	The first character of every line changes to uppercase; other characters are unaffected.

In addition to using the Change Case sheet, you can also select individual case change actions from the Change Case submenu immediately below the Change Case... command.

Shift Left / Shift Right

These commands indent or outdent the selected text by one tab stop. If there is no selection, this command works on the current line. Hold down the Shift key while choosing these commands, to have TextWrangler indent or outdent the text by one space instead of one tab stop.

TextWrangler also entabs and detabs on the fly as you shift text. For example, if the selected text is indented one tab stop and you apply Shift Left One Space, the tab will be converted to spaces and the text will be outdented one space. If you then apply Shift Right One Space, the spaces will be converted back to a single tab.

Un/Comment Selection

This command automates the task of commenting and uncommenting sections of code in various programming languages. Choose a range of text and apply this command to add or remove comments to it, depending on its initial comment state. If there is no selection, this command is disabled.

You can use the Options sheet of the Installed Languages list in the Languages preference panel to modify or set comment strings for any available languages.

Note If you have set custom comment delimiters for HTML in the Languages preference panel, those delimiters will be honored when you use the Un/Comment command. However, they will not affect the operation of the HTML-specific comment commands on the Markup menu.

Hard Wrap

This command wraps long lines by inserting hard line breaks and can reflow (fill) paragraphs if desired. See “How TextWrangler Wraps Text” on page 88 for more information.

Add Line Breaks

This command inserts a hard line break at the end of each line of text as displayed. See “How TextWrangler Wraps Text” on page 88 for more information.

Remove Line Breaks

This command removes carriage returns and spaces from sections of text. Use this command to turn text that has hard line breaks into text that can be soft-wrapped. See “How TextWrangler Wraps Text” on page 88 for more information.

Convert to ASCII

This command is no longer present; however, the “Replace with code” option of Zap Gremlins now performs the same conversions.

Educate Quotes

This command converts straight quotes (" and ') to typographer’s quotes (“ ” and ‘ ’).

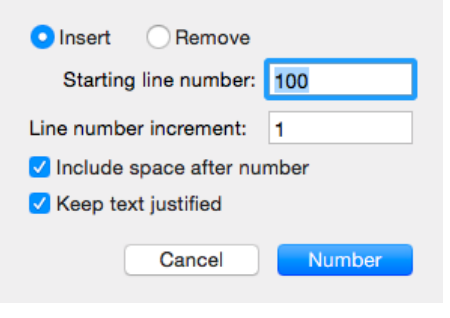
Note You should not use this plug-in to prepare text for use in a web page or an email, as typographer’s quotes in the Mac character set will generally not be properly displayed by applications on other platforms.

Straighten Quotes

This command performs the reverse of Educate Quotes; it converts typographer’s quotes (“ ” and ‘ ’) to straight quotes (" and ').

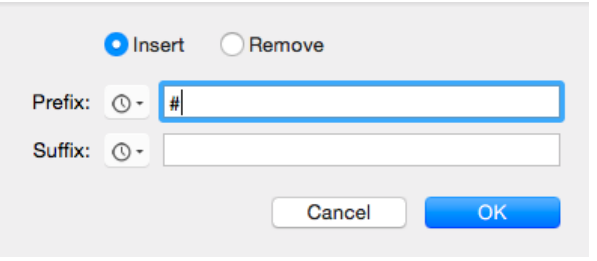
Add/Remove Line Numbers

This command displays a sheet which allows you to add or remove line numbers for each line of the selected text or of the document. You can set the starting number and numbering increment, as well as whether to include a trailing space, and whether to right-justify the inserted numbers, by choosing the appropriate options.



Prefix/Suffix Lines

This command displays a sheet which allows you to insert (or remove) the specified prefix and/or suffix strings on each line of the selected text or of the document.

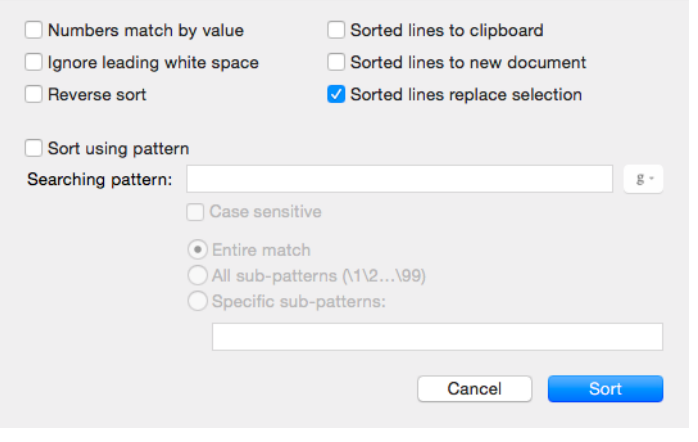


If you define both a prefix and a suffix string, TextWrangler will apply them to the text at the same time.

Note When using the "add prefix", "add suffix", "remove prefix", or "remove suffix" scripting commands, the string direct parameter is required.

Sort Lines

This command displays a sheet which allows you to sort lines of text by collating them in alphanumeric order. The sorted lines can be copied to the clipboard, be displayed in a new untitled window, replace the selection within the original document, or any combination of the three.



There are also options for ignoring white space at the beginning of lines, taking case distinctions into account, sorting strings of digits by numerical value instead of lexically, and sorting in descending rather than ascending order.

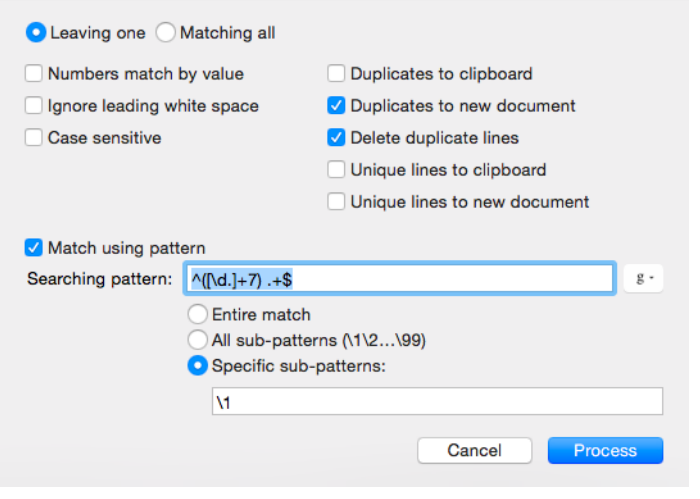
By checking the Sort Using Pattern option, you can specify a grep pattern to further filter the lines to be sorted. If the pattern contains subpatterns, you can use them to control the sort order based on the contents of the strings they match. When you sort using a grep pattern, the Case Sensitive option controls the case sensitivity of the pattern match in the same manner as the equivalent option in the Find dialog.

For example, suppose you are sorting a list of cities together with their two-letter state abbreviations, separated by a tab character. The pattern and subpatterns shown in the figure will sort the results first by city name and second by state abbreviation. Changing the contents of the Specific Sub-Patterns field from “\1\2” to “\2\1” will instead sort the results by state first and by city second.

IMPORTANT When you use a grep pattern with this command, matches are not automatically anchored to line boundaries, so ambiguous patterns may produce unpredictable results. To avoid this problem, you should use the line start ^ and line end operators as necessary. Also, keep mind that the pattern will only be tested against a single line at a time. So, if the pattern matches one or more sets of multiple lines within in the document, but does not match any individual lines, TextWrangler will not sort the contents of the document.

Process Duplicate Lines

This command displays a sheet which allows you to locate duplicate lines within a body of text and operates on them in various ways.



The Matching All option processes all duplicate lines; Leaving One ignores the first of each set of duplicate lines and processes only the additional ones.

The Numbers Match by Value and Ignore Leading White Space options allow you to choose whether strings of digits should be evaluated numerically or compared as strings, and whether white space at the beginnings of lines should be considered.

The Match Using Pattern option allows you to use a grep pattern to further filter the lines to be processed. You can enter a pattern in the Searching Pattern field, or choose a stored pattern from the popup menu. The Match Using: radio buttons control what part of the specified pattern should be used to determine duplication.

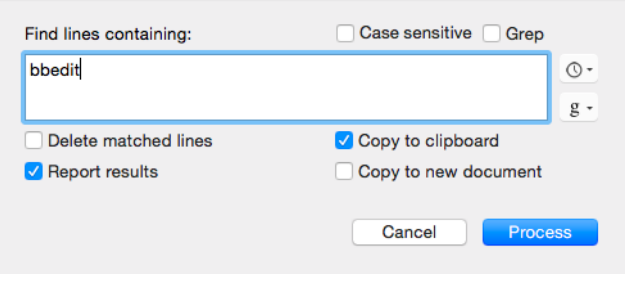
IMPORTANT When you use a grep pattern with this command, matches are not automatically anchored to line boundaries, so ambiguous patterns may produce unpredictable results. To avoid this problem, you should use the line start ^ and line end operators as necessary.

The options on the right-hand side of the sheet allow you to specify how duplicate lines should be handled once they have been identified. You can copy duplicate lines to the clipboard (Duplicates to Clipboard), copy them to a new document (Duplicates to New Document Window), and/or delete them from the current document (Delete Duplicate Lines). You can likewise specify how to handle the lines that are *not* duplicated by choosing Unique Lines to Clipboard and/or Unique Lines to New Document).

Since each of these options is an independent checkbox, you can select any combination of them that you wish. For example, selecting both Delete Duplicate Lines and Unique Lines ton Clipboard would delete the duplicate lines from the document and copy them to the clipboard for pasting elsewhere.

Process Lines Containing

This command displays a sheet which allows you to search the active window for lines containing a specified search string and then removes those lines or copies them to the clipboard. The options on the left side of the dialog box control how the search is performed and the options on the right side control what happens to the lines that are found.



To specify a search pattern, enter it in the Find Lines Containing field. If you do not want TextWrangler to match text when the letters in the text differ from the letters in the search string only by case (upper-case versus lower-case), select Case Sensitive.

To search using a grep pattern, select Use Grep and enter the pattern in the text field. You can also select a predefined search pattern from the Patterns popup menu.

Note If the selection ends in a trailing carriage return, the carriage return will be omitted from the search string copied into the text field.

The checkboxes on the right of the sheet control the way lines containing the specified search pattern will be processed. By selecting the appropriate combinations of these options, you can achieve the effect of applying various editing commands to each line:

- Setting both Copy to Clipboard and Delete Matched Lines on is equivalent to applying the Cut command.
- Setting Copy to Clipboard on and Delete Matched Lines off is equivalent to applying the Copy command.
- Setting Copy to Clipboard off and Delete Matched Lines on is equivalent to applying the Clear command.

The Copy to New Document option opens a new, untitled document containing copies of all lines matching the search pattern, whether or not they are deleted from the original window. By using this option and turning Copy to Clipboard off, you can collect all matching lines without affecting the previous contents of the clipboard.

The Report Results option causes TextWrangler to display a dialog reporting the total number of lines matched, regardless of their final disposition. With all of the other options turned off, this can be useful for pretesting the extent of a search operation without affecting the clipboard or the contents of the original window.

Increase and Decrease Quote Level

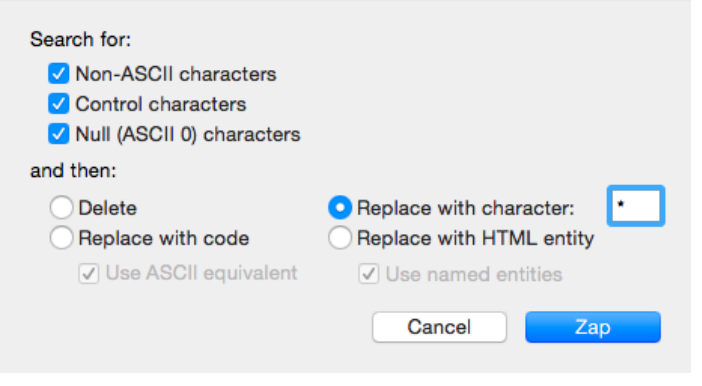
These commands respectively insert or delete a standard Internet quote character (“>”) from each line of the selected hard-wrapped text, or for the current line if there is no selection.

Strip Quotes

This command removes all Internet-style quoting from the selected hard-wrapped text, or from the current line if there is no selection.

Zap Gremlins

This command displays a sheet which allows you to remove or replace various non-printing characters, often known as “gremlins”. Use this command when you have a file that may contain extraneous control characters, or any non-ASCII characters, which you wish to identify or remove.



The checkboxes on the left-hand side of the sheet determine which types of characters the Zap Gremlins command affects, while the radio buttons on the right-hand side determine what to do with gremlins that are found.

Zap Non-ASCII characters

When this option is selected, Zap Gremlins zaps *all* characters in the file that do not fall in the 7-bit (or ASCII) range. Examples of such characters include special Macintosh characters such as bullets (•) and typographer's quotes (“ and ”, ‘ and ’), as well as all multi-byte characters. In general, such special characters are those that you type by holding down the Option key.

Zap Control characters

When this option is selected, Zap Gremlins zaps a specific range of invisible low-ASCII characters, also known as control characters. Control characters can cause compilers and other text-processing utilities to malfunction, and are therefore undesirable in many files.

Zap Null (ASCII 0) characters

When this option is selected, Zap Gremlins zaps all instances of the null character (ASCII 0). Like other control characters, nulls can cause many programming tools and text-processing utilities to malfunction. This specific option is included in case you want to remove only nulls without affecting other control characters that may be present in a file.

Delete

This option removes the zapped character completely from the text. It is useful if you are only interested in destroying gremlins and you do not care where they were in the text.

Replace with code

This option causes TextWrangler to handle each gremlin character according to its value as follows:

TextWrangler will convert certain eight-bit Mac Roman characters (characters whose decimal values are greater than 128 and less than 255) to 7-bit (printable ASCII range) equivalents. Converted characters include umlauted and accented vowels, ligatures, typographer's quotes, and various specialized punctuation forms. This conversion may entail expansion to multiple characters; for example, in the case of ligatures.

When the *Use ASCII equivalent* option is also enabled, TextWrangler will convert all extended Roman characters to their closest ASCII equivalent.

Otherwise, TextWrangler will convert all other gremlins present to escaped hexadecimal format. The escape code is formed via the same convention used by the C programming language: \0x followed by the character code in hexadecimal (base 16). This option is useful for identifying both the value and the location of gremlin characters. Later, you can search for occurrences of \0x to locate the converted characters. (Searching for the grep pattern of “\\0x..” will select the entire character code for easy modification or deletion.)

Replace with <character>

This option replaces the gremlin with the character you type in the text field next to the radio button. It is useful for identifying the location of gremlins, but not their value. The replacement character can be specified not only as any typeable character, but also by using any of the special characters defined for text searches, including hex escapes. (See “Special Characters” on page 120.)

Note In some cases, this option could be counterproductive, since hex escapes (\xNN) can themselves be used to insert unprintable characters.

Entab

This command displays a sheet which allows you to set the number of consecutive space characters which should be converted into tabs. This transformation is useful when you are copying content from many online sources, which use spaces to line up columns of text. If you do not use a monospaced font, columns usually will not line up unless you entab the text first.

Detab

This command displays a sheet which allows you to set the number of consecutive spaces which should replace each tab. This command is useful when you are preparing text for use in a program which has no concept of tabs as column separators, for email transmission, and similar purposes.

Normalize Line Endings

This command converts a document containing mixed line endings to have a uniform set of line endings.

If you open a file which contains a mixture of Mac, Unix, and DOS/Windows line endings, the “Translate Line Breaks” option may not suffice to properly convert the document for viewing and editing. After conversion, the document may appear to not have any line breaks at all (this usually happens if the first line break in the file is a Mac line break, and all the rest are Unix), or to have an invisible character at the beginning of each line.

Should this happen, use Normalize Line Breaks to convert the remaining line endings, and save the document. Once you have done this, the document’s line endings will be consistent, and TextWrangler’s line-break translation will suffice when you next open it.

CHAPTER

6

Windows & Palettes

This chapter describes the commands in the Window menu. These commands allow you to arrange and access editing and browser windows quickly, and also to access TextWrangler's tool and utility palettes.

In this chapter

Window Menu **113**
Minimize Window **113**
Bring All to Front **113**
Palettes **113**
Character Inspector – 114 • *Colors* – 114 • *Scripts* – 114
Text Filters – 114 • *Windows* – 114
Save Default <type of >Window **115**
Cascade Windows **115**
Arrange **115**
Cycle Through Windows **115**
Exchange with Next **115**
Synchro Scrolling **115**
Window Names **115**
Zoom (key equivalent only) **116**

Window Menu

The Window menu provides easy, centralized access to all of TextWrangler's tool and utility palettes, in addition to offering commands that you can use to access and organize editing and results windows on screen.

Minimize Window

This command puts the frontmost window into the Dock. Click the window icon in the Dock to restore the window. Hold down the Option key and this command becomes Minimize All Windows.

Bring All to Front

This command brings all un-minimized TextWrangler windows to the front.

Palettes

The Palettes submenu provides quick access to all of TextWrangler's numerous tool palettes and utility windows. Choosing an item from this submenu toggles display of the corresponding palette.

When moved or resized, palettes now automatically "snap" to the edges of the screen and the edges of other palettes. You can override this behavior by holding down the Shift key while dragging or resizing.

Character Inspector

This command opens a palette which displays the character values of the selected text in several standard formats. You may also select the displayed character values and either copy them or drag & drop them into a document.

Colors

This command opens the system color picker, which you can use to insert hex color values into source code, or HTML and XML files.

Scripts

The Scripts palette displays all the scripts currently installed in the Scripts subfolder of TextWrangler's application support folder. See Chapter 2, "Scripts", for more information about using scripts in TextWrangler.

Text Filters

The Text Filters palette displays all the text filters currently present in the Text Filters subfolder of TextWrangler's application support folder. See Chapter 2, "Text Filters", for more information about using text filters in TextWrangler.

Note The Set Shortcut button in the Scripts and Text Filters palettes allows you to assign a key equivalent to the currently selected item. You can use combinations of the Command, Shift, Option, and Control keys, plus any single other key, to create such equivalents, except that any equivalent must contain either the Command or Control keys (or both). You can also map Function keys directly to items, with or without the use of a modifier.

Windows

The Windows palette displays the names of all open windows ordered by name and kind. You can open a file by dragging its icon from the Finder into the Windows palette.

Document windows, which correspond to text files, have a document icon next to them; display windows, such as browsers and search results windows, do not. A solid diamond to the left of a window's name means that the window's contents have been modified and have not yet been saved, while a hollow diamond indicates that the window's state has been modified but not yet saved.

To bring any window to the front, click its name in the Windows palette. You can select one or more windows in the list, and choose the Save, Close, or Print commands from the action menu at the top of the palette. Holding down the Option key changes these commands to Save All, Close All, and Print All, which apply to all listed windows for which the given command is possible. You can also Control-click on any selected windows and apply the Save, Close, or Print commands from the resulting contextual menu.

"Hovering" the mouse over a window name displays a tool tip showing the full window title; this is useful for names that have been truncated with ellipses (…) because they are too long to fit within the width of the window. If you hold down the Option key, the tool tip will appear instantly, with no hovering delay. Holding down the Command key displays the full pathname for document windows (or other relevant windows such as disk browsers and FTP browsers).

Save Default <type of >Window

The Save Default Window command stores the position and size of the front window in TextWrangler's preferences, and TextWrangler will create all new windows of the same type with the stored position and size.

By default, new windows always stack down and right 20px. If you have saved a default window size, and that window is of full screen height, new windows will just stack to the right, and preserve their saved height.

Each type of window has its own default position and size. For instance, the default position and size for editing windows is different from the default position and size for disk browser windows.

Window position and size preferences are also keyed to the active screen configuration, so if you frequently switch screen layouts (as when connecting an external display to a portable), you can save separate default window preferences which will be applied depending on which screen configuration is active.

Cascade Windows

The Cascade Windows command cascades all open editing windows in the default fashion: each successive window will be moved incrementally down and to the right (as described above).

Arrange

The Arrange command now presents a submenu of multiple window arrangements to cascade, stack, or tile windows as indicated by their titles. You may choose any arrangement to immediately apply it, or assign a keyboard shortcut to any arrangment(s) that you use frequently in the "Menus & Shortcuts" preference panel.

Cycle Through Windows

This command sends the front window behind all the other windows. Hold the Shift key down when choosing this command to Cycle Through Windows Backwards, i.e. to bring the rearmost window to the front.

Exchange with Next

This command makes the second window the active window. Choose this command repeatedly to alternate between the front two windows.

Synchro Scrolling

When you have two or more windows open, Synchro Scrolling makes both files scroll when you scroll one. This feature is useful to look over two versions of the same file.

Window Names

The last items in the Window menu are the names of all the open documents, browsers, and other editing windows. Choose a window's name from this menu (or use its numbered Command key equivalent, if applicable) to bring that window to the front.

Tip You can also use the Windows palette to quickly select any open window.

Zoom (key equivalent only)

There is no longer a Zoom command in the Window menu, but the key equivalent Command-/still works. Zoom will produce the same effect as clicking a window's zoom box: it makes the active window larger if it is small, or returns it to its original size if it was previously enlarged by a Zoom command.

When zooming windows, TextWrangler will move the window as little as possible (consistent with maximizing the window's size). This behavior is similar to what the Finder does when zooming a window.

CHAPTER 7

Searching

This chapter describes TextWrangler's powerful Find and Multi-File Search commands, now enhanced with a flexible file filtering mechanism. It tells you how to search for text in the active window or within a set of files. TextWrangler can also do advanced pattern, or grep, searching. To learn about pattern searching, you should read this chapter first and then read Chapter 8, "Searching with Grep."

In this chapter

Search Windows 117
Basic Searching and Replacing 118
Search Settings – 120 • *Special Characters* – 120
Multi-File Searching 121
Find All and Multi-File Search Results – 123
Specifying the Search Set – 124
Multi-File Search Options – 126 • *File Filters* – 126
Searching SCM Directories – 128
Multi-File Replacing 129
Live Search 130
Search Menu Reference 130

Search Windows

TextWrangler's Find and Multi-file Search windows provide a consistent modeless interface to TextWrangler's powerful text search and replace capabilities.

If you are familiar with the modal Find dialog used in older versions, you'll generally feel at home, but there are some important differences and improvements of which you should be aware:

The Find dialog has been split in two, with a Find window for searching only the front document, and a Multi-File search window for searches which span more than one document, including folders, arbitrary open documents, results browsers from prior searches, etc.).

The set of search options which configure how text is actually searched (for single-file searches) has been condensed down to a single pair of options: "Selected text only" and "Wrap around".

- "Selected text only" affects **only** the Find All and Replace All operations: if there is a selection range in the front document, these operations will affect search only the contents of the selection range if this option is on, or the entire document (starting from the top) if this option is off.

- "Wrap around" affects **only** the "Next", "Previous", "Replace", and "Replace & Find" operations: if this option is on and the search reaches the end (or the beginning) of the document, then TextWrangler will continue the search from the appropriate end of the document.

Keyboard navigation is considerably different due to the Find and Multi-File Search windows' modeless nature.

- Pressing the Return or Enter key with focus in the Find field will perform "Next" in the Find window or "Find All" in the Multi-File Search window.
- Pressing the Escape key will close the window.
- Choosing an appropriate command in the Search menu will trigger the corresponding action in the front Find window.
- TextWrangler supports the following key equivalents to control (toggle) the search options contained in the Find and Multi-File Search windows. The factory default key equivalents for these options are as follows:

Case sensitive	Control-Shift-N
Entire word	Control-Shift-E
Grep	Control-Shift-G
Selected text only	Control-Shift-S
Wrap around	Control-Shift-W
Open search history	Control-Shift-H
Open saved patterns	Control-Shift-P

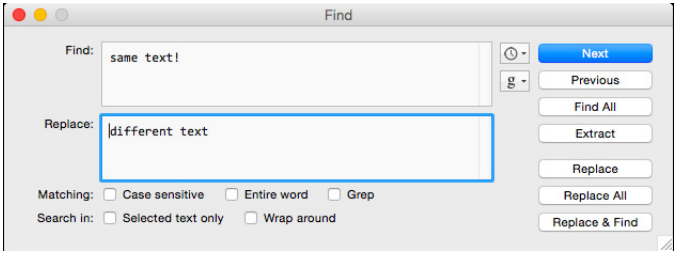
If these assignments overlap with any keyboard equivalents for clippings that you have set, or if you just wish to change them, you can do so via the "Find Windows" section of the Menus & Shortcuts preference panel.

Note The "Replace All" command replaces **all** occurrences of the search string within the document (or in the selection if there is one and "Search Selection Only" is checked). If you wish to replace only occurrences between the current insertion point and the end of the document, you can instead apply the Replace to End command in the Search menu.

Basic Searching and Replacing

This section describes the basic steps for searching and replacing text in a document. Later sections in this chapter cover more advanced techniques. To search and replace text in the front document, follow these steps:

1 **Choose Find from the Search menu. TextWrangler opens the Find window.**



2 **Type the string you are looking for in the Find text field.**

You can use special characters in the Find text field to search for tabs, line breaks, or page breaks. See “Special Characters” later in this section.

3 **Type the replace string (if any) in the Replace text field .**

TextWrangler persistently remembers the pairs of search and replace terms that you have most recently used. If you want to repeat a previous search or replace, you can choose the appropriate entry from the Search History popup menu at the right of the Find text field to fill in the Find and Replace fields.

Note The size of both the search and replace terms is limited only by available memory.

4 **Turn on any options that you want to apply to your search.**

For more info about these options, see “Search Settings” later in this section.

5 **Click one of the buttons along the right side of the dialog box.**

The following table explains what each of the buttons does.

This button…	Does this…
Next	Finds the first occurrence of the text in the active window after (below) the current insertion point.
Previous	Finds the first occurrence of the text in the active window before (above) the current insertion point.
Find All	Finds all the occurrences of the search string and displays the results in a search results browser.
Replace	Replaces the current selection with the replace string.
Replace All	Replaces every occurrence of the search string in the active window with the replace string.
Replace & Find	Replaces the current selection with the replace string, then finds the next occurrence of the text in the active window.

Once you have entered a search string (and also, if desired, a replace string), you can also use the commands in the Search menu to find and replace matches (see “Search Menu Reference” later in this chapter). The table below summarizes the most common commands you can use at this point.

This command…	Does this…
Find Next	Finds the next occurrence of the search string. To reverse the search direction, hold down Shift.
Replace	Replaces the selection with the replace string.
Replace All	Replaces all occurrences of the search string within the document with the replace string.

This command…	Does this…
Replace to End	Replaces every occurrence of the search string from the current insertion point to the end of the document with the replace string.
Replace & Find Again	Replaces the selection with the replace string and looks for the search string again.

Search Settings

The checkboxes in the Find window lets you control how TextWrangler searches your document for the indicated text.

Case Sensitive

When this checkbox is selected, TextWrangler treats upper- and lowercase letters as different letters. Otherwise, TextWrangler treats upper- and lowercase letters as if they were the same.

Entire Word

When this checkbox is selected, TextWrangler matches the search string only if it is surrounded in the document text by word-break characters (white space or punctuation). Otherwise, TextWrangler matches the search string anywhere in the text.

Grep

When this checkbox is selected, TextWrangler treats the search and replace strings as grep patterns. Otherwise, TextWrangler searches the document for text that matches the search string as it appears literally, and will replace any matched text with the replace string. To learn more about pattern searching see “Searching with Grep” on page 137.

Selected Text Only

When this checkbox is selected, TextWrangler searches only the selected text. Otherwise, TextWrangler searches the entire document.

Wrap Around

When this checkbox is selected, TextWrangler continues searching from the beginning of the document if a match is not found (or from the end of the document if searching backwards). Otherwise, TextWrangler stops searching when it reaches the end (or the beginning if searching backwards) of the file.

Special Characters

You can use the following special characters to search for line breaks and other non-printing characters, as well as hexadecimal escapes to search for any desired 8-bit character.

Character	Matches…
\r or \n	line break
\t	tab
\f	page break (form feed)

Character	Matches…
\xNN	hexadecimal character code NN (for example, \x0D for CR)
\x{*NNNN*}	any number of hexadecimal characters *NN…* (for example, \x{0} will match a null, \x{304F} will match a Japanese Unicode character)
\\	backslash (\)

The form of a hex escape is "\xNN", where "N" is any single hex digit [0-9,A-F]. The "x" may be upper or lower case. (You can use the ASCII Table in the Window menu to find the hex value for any 8-bit Macintosh character.) You can perform a literal search for any character, including a null, using this option. Malformed escapes are treated as literal strings.

Multi-File Searching

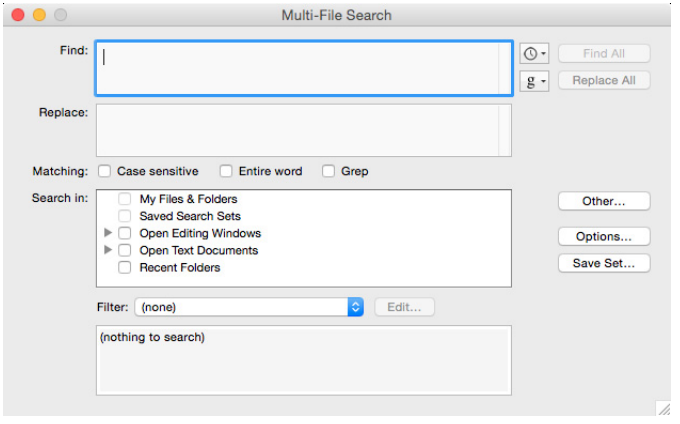
The main difference between single-file searching and multi-file searching is that to perform a multi-file search, you must specify the files to be searched. TextWrangler gives you a great deal of flexibility in how to do this. You can search all the files in a given folder, in open editing windows, or in an existing search results browser. For even greater control, you can select a diverse set of search sources, or apply TextWrangler's advanced file filtering capabilities.

When you start a search, TextWrangler will display a search progress window and return control, so that you can continue working. You can perform more than one multi-file searches at a time; each search will have its own progress window. Closing a search's progress window or clicking Cancel in the progress window will stop the operation, and TextWrangler will display a search results browser containing any matches found up to that point.

Starting a Search

To search for a string in multiple files, do the following steps:

1 **Choose Multi-File Search from the Search menu, or type Command-Shift-F, to open the Multi-File Search window (if it is not already open).**



2 **Type the string you are looking for in the Find text field.**

3 **Type the replace string (if any) in the Replace text field.**

Be sure to read the section "Multi-File Replacing" later in this chapter if you want to perform replace operations.

4 **Turn on any search options that you want to apply to your search.**

To learn more about these options, see "Search Settings" earlier in this chapter.

5 **Drag a folder to the search target area to search its contents, or select any of the available search sources in the Sources list to specify the set of files to search.**

See "Specifying the Search Set" later in this chapter for more information.

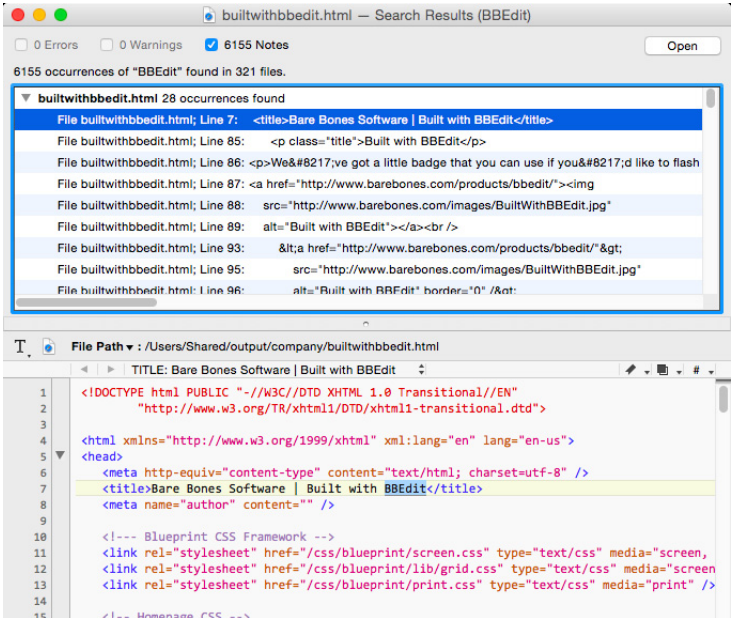
6 **Click one of the buttons along the right side of the dialog box to begin the search.**

The table below tells you what each of the buttons does.

This button…	Does this…
Find All	Finds all occurrences of the search string in all the files in the selected search sources. TextWrangler displays the results in a search results browser.
Replace All	Finds all occurrences of the search string in all the files in the selected search sources and replaces them with the replace string.
Options	Brings up the Search Options sheet.
Save Set	Creates an entry under the "Saved Search Sets" heading in the search sources list which you can later choose to reselect the same search sources.
Other	Select arbitrary file(s) or folder(s) to add to the search sources.

Find All and Multi-File Search Results

When you perform a Find All search, either on a single file or across multiple files, TextWrangler will open a search results browser which lists every occurrence of the search string in the selected file(s)



The information at the top of the window tells you how many matches TextWrangler found in the set of files you specified, as well as specifying whether there were any error conditions or warnings generated during the search. You can display or hide any combination of errors, warnings, and matches, by checking the appropriate options.

The middle panel lists each line that contains the matched text. Every match is identified by file name and line number.

Click any match in the list of occurrences to have TextWrangler display the part of the file that contains the match in the text pane.

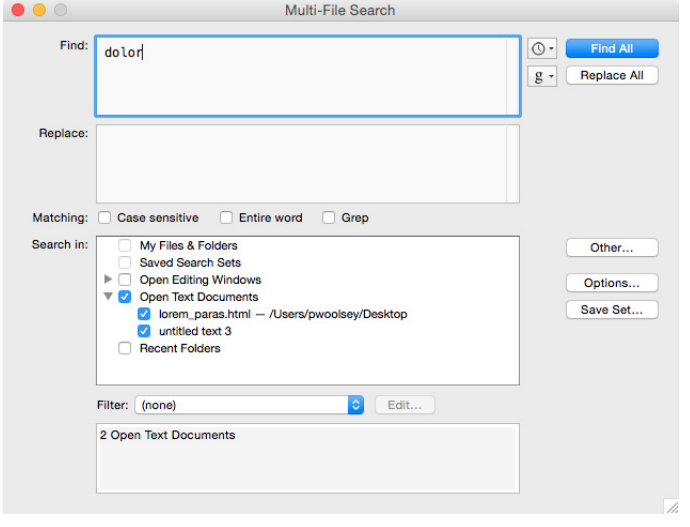
IMPORTANT You can edit text directly within a search results browser, or double-click any line that contains a match to open the corresponding file at the point of the match.

After you have opened a file, you can use the Find Again, Replace, Replace All, and Replace & Find Again commands in the Search menu to continue searching it, as if you had chosen a File by File search. See the next section for information on File by File searching.

Note You can use a search results window as the basis of another multi-file search. See "Specifying the Search Set" below.

Specifying the Search Set

To specify which files and folders TextWrangler should examine when performing a multi-file search, just select items from the Search In list of the Multi-File Search window.



You can choose multiple sources for a multi-file search, and you can mix different types of sources. Available sources include:

- specified individual files
- the files in any selected or recently-searched folder
- open text documents
- the files listed in any search results browser or compile errors browser
- the files and folders contained within any Zip archives
- any "Smart Folders" which you have saved in the Finder (TextWrangler will automatically list any such items present in the "Saved Searches" folder for your login account)

To add a file, folder, or other suitable item to the Search In list, click Other in the Multi-File Search window, and choose the item using the resulting selection sheet. (You can select multiple items to be added.)

To designate any item in the list as a search source, click on the box next to its name, or double-click on the name, to add a checkmark. To deselect a search source, click the box next to its name, or double-click its name, to turn off the checkmark. To select a single source only, and deselect all other sources, Command-click on the checkbox next to the desired source's name. To remove a search source from the list, click on the minus sign (–) to the right of its name. (Doing so removes only the entry from the list, **not** the original item.)

TextWrangler will display a summary of the selected sources in the information box at the bottom of the Multi-File Search window. Here are some common scenarios.

Searching the files in a folder

To search the files in a folder, click on the box next to the folder's name, or double-click its name, in the Sources list. If the folder you want to search is not in the Sources list, click the Other button at the right of the dialog and pick the folder using the resulting selection sheet.

You can also drag a folder from the Finder directly into the search items box of the Find & Replace dialog to choose it as the source.

Note The Choose a Folder dialog will display any packages it encounters as folders (rather than just as single files, the way they appear in the Finder). This allows you to navigate their internal structure just as you would any other folder. Similarly, you can drag a package from the Finder into the path box in the Find & Replace dialog and it will be treated as a true folder rather than as a single file.

Searching all open documents

You can choose any or all open text documents as search sources. This option allows you to search documents that have not yet been saved to a file, or which contain unsaved changes. To choose all open documents, click the box next to the Open Text Documents item, or double-click on the item in the list.

Searching the contents of compressed archives

You can control whether TextWrangler should search within the contents of compressed archives (".bz2", ".gz", and ".zip") via the "Search compressed files" option in the Multi-File Search window's "Options" sheet. When this option is off, TextWrangler will skip all bz2, gz, and Zip while searching, even if they may contain compressed text files.

Searching the files contained in a results browser

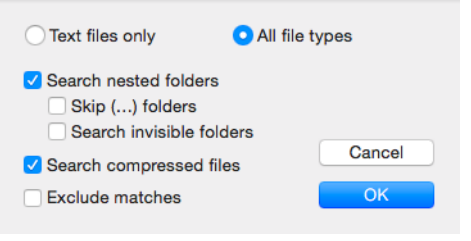
If a previous multi-file search found many files that contain your search string, you may want to narrow the search. To search the files listed in any results browser window, click the box next to that browser's name, or double-click on its name, in the Sources list. You can also click the box next to the Results Browsers item, or double-click on this item, to search the files listed in all results browsers.

Saved Search Sources

You can use the Saved Search Sources popup menu to store specific sets of search sources for later reuse. To save a set of search sources, choose Remember this Set from the popup menu and give the set a name in the resulting dialog. To select a saved set of search sources, choose that set's name from the pop-menu.

Multi-File Search Options

Click the Options button to display the search options sheet.



To search the contents of all subfolders within the folders you choose, select the Search Nested Folders option in the resulting sheet. You can also choose to skip any folders whose names are enclosed in parentheses here by selecting the Skip (…) Folders option, or whether to search the contents of invisible folders by selecting the Search Invisible Folders option.

You can also choose to search only text files or to search all file types. If you have image files or other non-text files in search source folders, it may be a good idea to restrict the search to only text files. This setting is applied in addition to any file filter (see next section) and in fact takes effect *before* the filter.

To find only files whose contents do **not** contain the search string, select the Exclude Matches option.

You can further restrict which files from the chosen sources will be searched by applying a file filter. See "File Filters" (below) for more details.

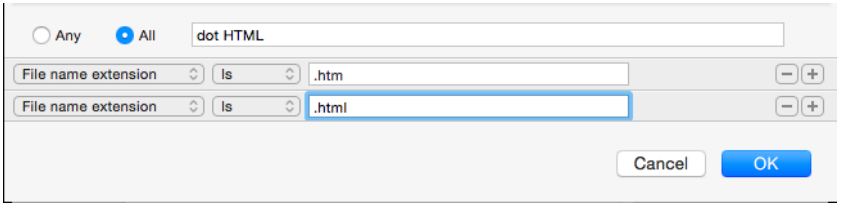
File Filters

If you do not want to search every file in the set you selected, but want to include only those that meet certain criteria (such as those created on a certain date, or only those created by TextWrangler and not some other program, or those that are HTML or Perl documents), you can use a file filter.

To apply a file filter, just choose it from the Filters popup menu in the Multi-File Search window. If none of the available filters meets your needs, you can define a new one, or create a temporary filter.

New Filter

To define a new saved file filter, select New… from the popup menu. TextWrangler will ask you for a filter name, and then display the Edit Filter dialog (below). You can also define new file filters in the Filters panel of the Setup window (see page 194).



Note If the Setup window is open, any filters you define in the Multi-File Search window will not be available in the Filters panel of the Setup window until you close and reopen the Setup window.

The Edit Filter dialog lets you specify multiple criteria that determine whether a given file is selected by the filter. You can choose whether these criteria are exclusive (that is, whether a file must meet *every* listed test to be selected) or inclusive (that is, whether a file that meets *any* of the tests is selected) using the Every (AND) and Any (OR) radio buttons at the top of the dialog.

To add a test, click the Add (+) button, and a new row will appear in the dialog.

Within each row (criterion), the left-hand popup lets you specify which attribute of a file you wish to test. TextWrangler lets you test a file's name, the name of its enclosing folder, its creator or type, its creation and modification date (or both date and time), or its Finder label, visibility, or the programming or markup language it is written in. You can also test the content of a file, using the "Contents" criterion.

The center popup lets you choose the test to be applied to the selected attribute. The available options here change depending on what attribute you selected. If you choose Visibility in the first column, for instance, your only choices are whether the file is or is not visible, However, if you choose File Name in the first column, the middle column lets you test to see if the name does or does not exactly match, contain, begin with, or end with a particular string. You can also test file names to see if they match wildcard or Grep patterns.

Note In wildcard patterns, the asterisk (*) and question mark (?) characters have special meanings. The asterisk matches any number of characters, such that "*.c" matches any file whose name ends with ".c". The question mark matches a single character, so that "foo?" matches "food", fool", "foot", and many other words. Both the asterisk and the question mark can be used anywhere in a wildcard pattern, and any number of either can be used in a single pattern.

Grep patterns, also known as regular expressions, are a powerful method of selecting file names based on classes of text or repeating text. They are covered in great detail in the next chapter.

The right-hand text field specifies the match criterion. For example, when filtering by File Name, you type the text you want the name to match, contain, begin with, or end with (or not). When filtering by Language, you choose a supported language from a popup menu.

You can add any number of criteria using the Add (+) button. To delete any criterion, click the Remove (-) button next to it.

Click Save to save the file filter and use it for this search. TextWrangler will ask you to name the filter, and it will then appear in the Filters popup menu in the Find & Replace dialog (and in the Filter panel of the Setup window). Click Cancel to discard any changes you have made to the filter. (Hold the Option key when you click Cancel to skip the confirmation alert.)

Filtering by Name

In order to provide the greatest possible flexibility, TextWrangler offers several different criteria for filtering based on file names

File Name: Tests the complete string corresponding to the file name.

File Name Root: Tests only the "root" portion of the file name. Given a name of the form "foo.txt", the root is the string which occurs before the period, in this case "foo".

File Name Extension: Tests only the file name extension. In the above example, the extension is "txt". (Note that the extension does not include a period.)

Temporary Filters

Choose "(current criteria)" from the popup menu in the Find & Replace dialog to reuse the last set of criteria applied (either from using a saved filter, or from using the Edit button to define criteria). Thus, you can use filter criteria on the fly, without the need to create and store a throwaway filter.

Editing and Deleting Filters

To edit a file filter you have already defined, choose it from the Filters popup menu, change it as desired, and click Save. Since each filter must have a unique name, saving it will replace the old version of the filter. To delete a filter entirely, visit the Filters panel of the Setup window. (You can also create or modify filters there.)

Searching SCM Directories

When scanning folders for various purposes (multi-file search, Find Differences, and other batch operations), TextWrangler ignores all directories which contain administrative data for source-control management (SCM) tools: CVS, .svn, .git, .hg, .bzr. This behavior prevents any inadvertent modifications to such data which might otherwise occurs during a multi-file search or other batch operation. If you must search the contents of such directories, you can enable TextWrangler to do so by issuing the following Terminal command:

```
defaults write com.barebones.textwrangler
SkipSCMAdminDirsWhenScanningFolders -bool NO
```

Note The "Search Invisible Folders" option no longer enables TextWrangler to search within such directories.

Multi-File Replacing

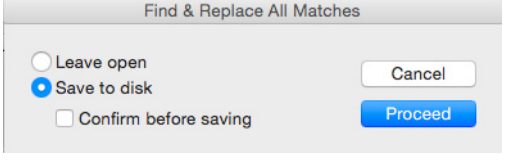
If you want to replace only some occurrences of text in multiple files, you can simply search those files, select the instances you want to change in the search results browser to open the files to those points, and perform the replacements individually. However, TextWrangler can also change *all* occurrences of a string in a group of files with one command.

Globally replacing text in more than one file works the same as replacing it in a single file. The only possible complication is that, if you make a mistake, it can have much wider consequences. If you are not sure what effect a replace operation will have, test it out on a few sample files (or a copy of your data) first!

To do a multi-file search and replace, replacing all occurrences:

1. **Enter your desired find and replace strings in the Multi-File Search window as described in the section "Multi-File Search."**
2. **Choose the files to be searched as described in "Specifying the Search Set".**
3. **To start the operation, click Replace All in the Multi-File Search window, choose the Replace All command, or type its key equivalent of Command-Option-R.**

   TextWrangler displays the Find & Replace All Matches dialog box:



This is what each of its options does:

This option…	Replaces all occurrences of the search string with the replace string and…
Leave Open	Leaves all the files open so that you can inspect the replacements. If there are many files that contain the search string, TextWrangler may run out of memory.
Save to Disk	Saves each file with the changes. When the Confirm Saves setting is active, you will have an opportunity to approve the changes before TextWrangler saves them to disk. You should *not* turn this off unless you are sure that the replace operation being done is what you want.
Show Results	Opens a results browser listing each of the files which was changed, and the number of changes in each file.

Live Search

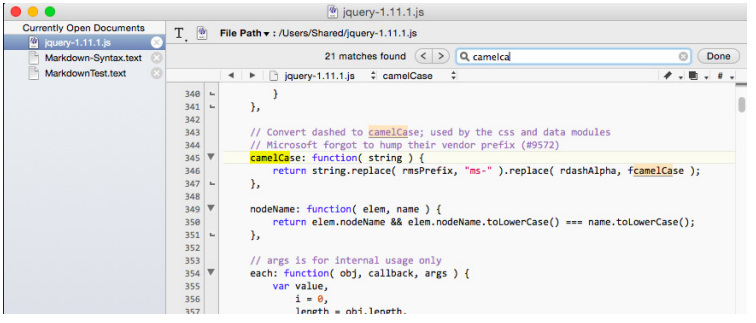
The Live Search command performs an incremental search. In other words, it shows the matching text as you type the search string, so you only have to type until you find the text you want.

Live Search always searches in the text view of the frontmost window; if that window has no text view, the Live Search command will be disabled.

To use Live Search:

1 **Choose Live Search from the Search menu, or type Command-Option-F.**

2 **Type the string you are looking for into the Live Search field.**



As you type, TextWrangler selects the first occurrence of what you have typed so far.

3 **To find the next occurrence of the matching text, click the Next (right) arrow, or type Return or Enter.**

4 **To find the previous occurrence of the matching text, click the Previous (left) arrow, or type Shift-Return or Shift-Enter.**

If Emacs key bindings are enabled, you can also type Control-S to start a Live Search, and then type Control-S or Control-R to search forward or backward respectively.

To clear the most recent word of the search string, you can type Option-Delete, or click on the "delete" button (the "X") within the search field to delete the entire search string.

To cancel Live Search, you may click the "Done" button in the search bar or type the Escape key.

Note The Live Search bar replaces the Quick Search window present in older versions.

Search Menu Reference

This section describes all of the commands in the Search menu.

Find

Opens the Find window (or the Find & Replace dialog). See "Basic Searching and Replacing" on page 118.

Multi-File Search

Opens the Multi-File Search window. See “Multi-File Searching” on page 121 and “Multi-File Replacing” on page 129.

Search in [Document’s Folder]

If a text document is active in the front window and that document is associated with a file on disk, this command will be enabled, and selecting it will open the Multi-File Search window with the search source pre-set to the parent folder of the document’s file, and that folder’s name and path will appear in this command’s menu name.

Search in … (Disk or Results Browser)

If the frontmost window is an editing window, this command’s name will reflect the name of the current document’s parent folder (if any).

If the frontmost window is a disk browser or results browser, this command's name will reflect the name of the directory currently visible in the disk browser, or the name of the results browser window.

Choosing this command will open the Multi-File Search window with the described target (the current document’s parent folder, or the selected directory) set as the search source.

If the Multi-File Search window is frontmost, this command will target the disk browser or results browser which is closest to the front (Z-order).

This command is also available in the Action (‘gear’) menu of disk browsers.

Live Search

Opens the Live Search bar. You can use this feature to interactively search for text strings, as described in the previous section.

Find Next/Previous

Searches the current document for the next occurrence of the search string. Hold down the Shift key to find the previous occurrence.

Find All

Finds all instances of the search string in the current document or search set, and displays a search results browser.

Find Selected Text/Previous Selected Text

Uses the selected text as the search string and finds the next occurrence of the selected text. Hold down the Shift key to find the previous occurrence of the selected text.

When you invoke this command, TextWrangler will add the current search string to its Search History list of recently-used search strings.

Tip You can also hold down the Option and Command keys as you double-click on a selection to search for the next occurrence of the selected text.

Use Selection for Find

Sets TextWrangler's search string to the currently selected text but does not perform a search. When you invoke this command, TextWrangler will add the current search & replace strings to its Search History list.

Use Selection for Find (grep)

When you hold down the Shift key, Use Selection for Find becomes Use Selection for Find (grep). This command sets TextWrangler's search string to the currently selected text and turns on the Grep option, but does not perform a search. When you invoke this command, TextWrangler will add the current search & replace strings to its Search History list.

Use Selection for Replace

Sets TextWrangler's replace string to the currently selected text but does not perform a search operation. When you invoke this command, TextWrangler will add the current search & replace strings to its Search History list.

Use Selection for Replace (grep)

When you hold down the Shift key, Use Selection for Replace becomes Use Selection for Replace (grep). This command sets TextWrangler's replace string to the currently selected text and turns on the Grep option, but does not perform a search operation. When you invoke this command, TextWrangler will add the current search& replace strings to its Search History list.

Replace

Replaces the selected text (usually an occurrence of the search string) with the replace string.

Replace All

Replaces **all** occurrences of the search string in a file with the replace string, or, starts a multi-file search & replace operation.

Replace to End

Replaces each occurrence of the search string from the current insertion point (or the start of the current selection range) to the end of the document.

Replace & Find Again

Replaces the selected text with the replace string and searches for the next occurrence of the search string.

Find Differences

Finds the differences between two files, or all of the files contained in two folders. See "Comparing Text Files" in Chapter 4 for more details.

Compare Two Front Documents

Performs a Find Differences on the two frontmost text documents, using the same settings currently active for the Find Differences command.

Compare Against Disk File

Performs a Find Differences between the contents of the front document and the disk file for that same document. This capability makes it easy to locate in-progress changes to a document.

Compare Against Previous Version

(Only on OS X 10.7 or later) Provides access to previous versions of the current document using the system's built-in file versioning capability. You can compare and integrate changes from any prior version into the current document as desired.

Apply to New

Applies the currently selected difference to the "New" version of two files which are being compared. See "Comparing Text Files" for more details.

Apply to Old

Applies the currently selected difference to the "Old" version of two files which are being compared. See "Comparing Text Files" for more details.

Compare Again

Find the differences between two files, using the same settings that were used in the last time you used the Find Differences command. See "Comparing Text Files" for more details.

Find in Reference

Performs a search for the selected symbol using an appropriate language-specific online resource. As for Find Definition, if there is no selection, TextWrangler will attempt to determine the symbol name by inspection around the insertion point.

For example, Find in Reference in a PHP document will look up the selected symbol on php.net; in a Ruby document, it will use the 'ri' interactive reference; in a Unix Shell Script, it will open the appropriate Unix man page.

For languages which don't have a pre-defined resources, lookups will performed on the Apple Developer Connection web site.

You can modify the URL template which TextWrangler uses to perform the lookup for a particular language by bringing up the Options sheet for that language in the Languages preference panel and editing the template directly. In the template, use "__SYMBOLNAME__" to indicate where the selected symbol name should be placed in the lookup string.

Go Menu Reference

This section describes all of the commands in the Go menu.

Line Number

When you choose this command, TextWrangler opens the Line Number sheet. Type in a line number and the frontmost text window will jump to display that line.

This command does not follow the usual convention of applying the last-used setting when invoked with the Option key pressed. Instead, if you select a number within the current document, then hold down the Option key and choose “Line Number”, TextWrangler will go directly to the correspondingly numbered line.

Note This command honors the “Use ‘Hard’ Line Numbering in Soft-Wrapped Text Views” option in the Editing preference panel.

Center Line

Will move the insertion point to the beginning of the middle or center line of the displayed text.

Named Symbol

When you choose this command, TextWrangler opens a sheet displaying all named symbols within the current document, i.e. all available functions and markers. You can navigate this list with the arrow keys, or type to filter the displayed list of symbols.

Functions

The following four commands will be available in any document which contains recognizable functions.

Reveal Start/End

When you choose one of these commands, TextWrangler will move the insertion point to a position immediately before the start or immediately after the end of the current function, where a function is any item which appears on the function popup menu. If you anticipate using these commands often, you may wish to assign them key equivalents in the Menus & Shortcuts preference panel.

Go to Previous/Next

When you choose one of these commands, TextWrangler will select the name of the previous or next function in the document, where a function is any item which appears on the function popup menu. If you anticipate using these commands often, you may wish to assign them key equivalents in the Menus & Shortcuts preference panel.

Markers

When you choose this command, TextWrangler opens a floating window which lists any markers associated with the active document. (For information about setting markers, see “Using Markers” on page 98). You can filter the list by typing a partial marker name into the search box.

Jump Points

The following commands are available in any document.

Previous

When you choose this command, TextWrangler will go to the last selection you made in the document which was outside the current view (an automatic jump mark), or the last location you marked with the Set command (see “Set” on the preceding page). If the current document does not contain any jump marks, this command is disabled.

Next

When you choose this command after navigating to an earlier jump mark, TextWrangler will go to the next later jump mark, or return to the most recent position of the insertion point. If you have not jumped back to a jump mark, this command is disabled.

Set

Choose this command to define the current insertion point location or selection range as a manual jump mark within the active document. You can navigate to jump marks using the Jump Back and Jump Forward commands.

CHAPTER

8 Searching with Grep

This chapter describes the Grep option in TextWrangler's Find command, which allows you to find and change text that matches a set of conditions you specify. Combined with the multi-file search and replace features described in Chapter 7, TextWrangler's grep capabilities can make many editing tasks quicker and easier, whether you are modifying Web pages, extracting data from a file, or just rearranging a phone list.

In this chapter

What Is Grep or Pattern Searching? **138**
Writing Search Patterns **138**
Most Characters Match Themselves – 138
Escaping Special Characters – 138
Wildcards Match Types of Characters – 140
Character Classes Match Sets or Ranges of Characters – 142
Matching Non-Printing Characters – 143
Other Special Character Classes – 144
Quantifiers Repeat Subpatterns – 145
Combining Patterns to Make Complex Patterns – 146
Creating Subpatterns – 146 • *Using Backreferences in Subpatterns* – 147
Using Alternation – 148 • *The "Longest Match" Issue* – 148
Non-Greedy Quantifiers – 149
Writing Replacement Patterns **150**
Subpatterns Make Replacement Powerful – 150
Using the Entire Matched Pattern – 150
Using Parts of the Matched Pattern – 151
Case Transformations – 152
Examples **153**
Matching Identifiers – 153 • *Matching White Space* – 153
Matching Delimited Strings – 154 • *Marking Structured Text* – 154
Marking a Mail Digest – 155 • *Rearranging Name Lists* – 155
Advanced Grep Topics **155**
Matching Nulls – 156 • *Backreferences* – 156
POSIX-Style Character Classes – 157
Non-Capturing Parentheses – 158
Perl-Style Pattern Extensions – 159 • *Comments* – 159
Pattern Modifiers – 160 • *Positional Assertions* – 161
Conditional Subpatterns – 163 • *Once-Only Subpatterns* – 164
Recursive Patterns – 166
Writing Search Patterns **138**

What Is Grep or Pattern Searching?

Grep patterns offer a powerful way to make changes to your data that "plain text" searches simply cannot. For example, suppose you have a list of people's names that you want to alphabetize. If the names appear last name first, you can easily put these names in a TextWrangler window and use the Sort tool. But if the list is arranged first name first, a simple grep pattern can be used to put the names in the proper order for sorting.

A grep pattern, also known as a regular expression, describes the text that you are looking for. For instance, a pattern can describe words that begin with C and end in l. A pattern like this would match "Call", "Cornwall", and "Criminal" as well as hundreds of other words.

In fact, you have probably already used pattern searching without realizing it. The Find window's "Case sensitive" and "Entire word" options turn on special searching patterns. Suppose that you are looking for "corn". With the "Case sensitive" option turned off, you are actually looking for a pattern that says: look for a C or c, O or o, R or r, and N or n. With the "Entire word" option on, you are looking for the string "corn" only if it is surrounded by white space or punctuation characters; special search characters, called metacharacters, are added to the search string you specified to indicate this.

What makes pattern searching counterintuitive at first is how you describe the pattern. Consider the first example above, where we want to search for text that begins with the letter "C" and ends with the letter "l" with any number of letters in between. What exactly do you put between them that means "any number of letters"? That is what this chapter is all about.

Note Grep is the name of a frequently used Unix command that searches using regular expressions, the same type of search pattern used by TextWrangler. For this reason, you will often see regular expressions called "grep patterns," as TextWrangler does. They're the same thing.

Writing Search Patterns

This section explains how to create search patterns using TextWrangler's grep syntax. For readers with prior experience, this is essentially like the syntax used for regular expressions in the Perl programming language. (However, you *do not* need to understand anything about Perl in order to make use of TextWrangler's grep searching.)

Most Characters Match Themselves

Most characters that you type into the Find window match themselves. For instance, if you are looking for the letter "t", Grep stops and reports a match when it encounters a "t" in the text. This idea is so obvious that it seems not worth mentioning, but the important thing to remember is that these characters *are* search patterns. Very simple patterns, to be sure, but patterns nonetheless.

Escaping Special Characters

In addition to the simple character matching discussed above, there are various special characters that have different meanings when used in a grep pattern than in a normal search. (The use of these characters is covered in the following sections.)

However, sometimes you will need to include an exact, or literal, instance of these characters in your grep pattern. In this case, you must use the backslash character \ before that special character to have it be treated literally; this is known as “escaping” the special character. To search for a backslash character itself, double it \\ so that its first appearance will escape the second.

For example, perhaps the most common “special character” in grep is the dot: “.”. In grep, a dot character will match any character except a return. But what if you only want to match a literal dot? If you escape the dot: “\.”, it will only match another literal dot character in your text.

So, most characters match themselves, and even the special characters will match themselves if they are preceded by a backslash. TextWrangler’s grep syntax coloring helps make this clear.

Note When passing grep patterns to TextWrangler via AppleScript, be aware that both the backslash and double-quote characters have special meaning to AppleScript. In order to pass these through correctly, you must escape them in your script. Thus, to pass \r for a carriage return to TextWrangler, you must write \\r in your AppleScript string.

Wildcards Match Types of Characters

These special characters, or metacharacters, are used to match certain types of other characters:

Wildcard	Matches…
.	any character except a line break (that is, a carriage return)
^	beginning of a line (unless used in a character class)
$	end of line (unless used in a character class)

Being able to specifically match text starting at the beginning or end of a line is an especially handy feature of grep. For example, if you wanted to find every instance of a message sent by Patrick, from a log file which contains various other information like so:

```
From: Rich, server: barebones.com
To: TextWrangler-Talk, server: lists.barebones.com
From: Patrick, server: example.barebones.com
```

you could search for the pattern:

```
^From: Patrick
```

and you will find every occurrence of these lines in your file (or set of files if you do a multi-file search instead).

It is important to note that ^ and $ do not actually match return characters. They match zero-width *positions* after and before returns, respectively. So, if you are looking for “foo” at the end of a line, the pattern “foo$” will match the three characters “f”, “o”, and “o”. If you search for “foo\r”, you will match the same text, but the match will contain four characters: “f”, “o”, “o”, and a return.

Note ^ and $ do not match the positions after and before soft line breaks.

You can combine ^ and $ within a pattern to force a match to constitute an entire line. For example:

```
^foo$
```

will only match “foo” on a line by itself, with no other characters. Try it against these three lines to see for yourself:

```
foobar
foo
fighting foo
```

The pattern will only match the second line.

Other Positional Assertions

TextWrangler's grep engine supports additional positional assertions, very similar to ^ and $.

Escape	Matches
\A	only at the beginning of the document (as opposed to ^, which matches at the beginning of the document and also at the beginning of each line)
\b	any word boundary, defined as any position between a \w character and a \W character, in either order
\B	any position that is *not* a word boundary
\z	at the end of the document (as opposed to $, which matches at the end of the document and also at the end of each line)
\Z	at the end of the document, or before a trailing return at the end of the doc, if there is one

Examples (the text matched by the pattern is underlined)

Search for: `\bfoo\b`

Will match: `bar foo bar`

Will match: `foo bar`

Will not match: `foobar`

Search for: `\bJane\b`

Will match: `Jane's`

Will match: `Tell Jane about the monkey.`

Search for: `\Afoo`

Will match: `foobar`

Will not match: `This is good foo.`

Character Classes Match Sets or Ranges of Characters

The character class construct lets you specify a set or a range of characters to match, or to ignore. A character class is constructed by placing a pair of square brackets […] around the group or range of characters you wish to include. To exclude, or ignore, all characters specified by a character class, add a caret character ^ just after the opening bracket [^…]. For example:

Character Class	Matches
`[xyz]`	any one of the characters x, y, z
`[^xyz]`	any character except x, y, z
`[a-z]`	any character in the range a to z

You can use any number of characters or ranges between the brackets. Here are some examples:

Character Class	Matches
`[aeiou]`	any vowel
`[^aeiou]`	any character that is not a vowel
`[a-zA-Z0-9]`	any character from a-z, A-Z, or 0-9
`[^aeiou0-9]`	any character that is neither a vowel nor a digit

A character class matches when the search encounters any *one* of the characters in the pattern. However, the contents of a set are only treated as separate characters, not as words. For example, if your search pattern is [beans] and the text in the window is “lima beans”, TextWrangler will report a match at the “a” of the word “lima”.

To include the character] in a set or a range, place it immediately after the opening bracket. To use the ^ character, place it anywhere except immediately after the opening bracket. To match a dash character (hyphen) in a range, place it at the beginning of the range; to match it as part of a set, place it at the beginning or end of the set. Or, you can include any of these character at any point in the class by escaping them with a backslash.

Character Class	Matches
`[]0-9]`	any digit or]
`[aeiou^]`	a vowel or ^
`[-A-Z]`	a dash or A - Z

Character Class	Matches
`[--A]`	any character in the range from - to A
`[aeiou-]`	any vowel or -
`[aei\-ou]`	any vowel or -

Character classes respect the setting of the Case Sensitive checkbox in the Find window. For example, if Case Sensitive is on, [a] will only match “a”; if Case Sensitive is off, [a] will match both “a” and “A”.

Matching Non-Printing Characters

As described in Chapter 7 on searching, TextWrangler provides several special character pairs that you can use to match common non-printing characters, as well as the ability to specify any arbitrary character by means of its hexadecimal character code (escape code). You can use these special characters in grep patterns as well as for normal searching.

For example, to look for a tab or a space, you would use the character class [\t] (consisting of a tab special character and a space character).

Character	Matches
\r	line break (carriage return)
\n	Unix line break (line feed)
\t	tab
\f	page break (form feed)
\a	alarm (hex 07)
\cX	a named control character, like \cC for Control-C
\b	backspace (hex 08) *(only in character classes)*
\e	Esc (hex 1B)
\x*NN*	hexadecimal character code *NN* (for example, \x0D for CR)
\x{*NNNN*}	any number of hexadecimal characters *NN…* (for example, \x{0} will match a null, \x{304F} will match a Japanese Unicode character)
\\	backslash

Use \r to match a line break in the middle of a pattern and the special characters ^ and $ (described above) to “anchor” a pattern to the beginning of a line or to the end of a line. In the case of ^ and $, the line break character is not included in the match.

Other Special Character Classes

TextWrangler uses several other sequences for matching different types or categories of characters.

Special Character	Matches
\s	any whitespace character (space, tab, carriage return, line feed, form feed)
\S	any non-whitespace character (any character not included by \s)
\w	any word character (a-z, A-Z, 0-9, _, and some 8-bit characters)
\W	any non-word character (all characters not included by \w, including carriage returns)
\d	any digit (0-9)
\D	any non-digit character (including carriage return)

A “word” is defined in TextWrangler as any run of non-word-break characters bounded by word breaks. Word characters are generally alphanumeric, and some characters whose value is greater than 127 are also considered word characters.

Note that any character matched by \s is by definition not a word character; thus, anything matched by \s will also be matched by \W (but not the reverse!).

Quantifiers Repeat Subpatterns

The special characters *, +, and ? specify *how many times* the pattern preceding them may repeat. {}-style quantifiers allow you to specify exactly how many times a subpattern can repeat. The preceding pattern can be a literal character, a wildcard character, a character class, or a special character.

Pattern	Matches
`p*`	zero or more p's
`p+`	one or more p's
`p?`	zero or one p's
`p{`*`COUNT`*`}`	match exactly *COUNT* p's, where *COUNT* is an integer
`p{`*`MIN`*`,}`	match at least *MIN* p's, where *MIN* is an integer
`p{`*`MIN,MAX`*`}`	match at least *MIN* p's, but no more than *MAX*

Note that the repetition characters `*` and `?` match *zero or more* occurrences of the pattern. That means that they will *always* succeed, because there will always be at least zero occurrences of any pattern, but that they will not necessarily select any text (if no occurrences of the preceding pattern are present).

For this reason, when you are trying to match more than one occurrence, it is usually better to use a + than a *, because + *requires* a match, whereas * can match the empty string. Only use * when you are sures you really mean "zero or more times," not just "more than once."

Try the following examples to see how their behavior matches what you expect:

Pattern	Text	Matches
`.*`	Fourscore and seven years	Fourscore and seven years
`[0-9]+`	I've been a loyal member since 1983 or so.	1983
`\d+`	I've got 12 years on him.	12
`A+`	BAAAAAAAB	AAAAAAA
`A{3}`	BAAAAB	AAA *(first three A's)*
`A{3,}`	BAAAAB	AAAA
`A{1,3}`	BAAAAB	AAA on the first match, the remaining A on the second match
`c?andy`	andy likes candy	"andy" on the first match, "candy" on the second
`A+`	Ted joined AAA yesterday	"AAA" on the first match; "a" from "yesterday" on the second

Combining Patterns to Make Complex Patterns

So far, the patterns you have seen match a single character or the repetition of a single character or class of characters. This is very useful when you are looking for runs of digits or single letters, but often that is not enough.

However, by combining these patterns, you can search for more complex items. As it happens, you are already familiar with combining patterns. Remember the section at beginning of this discussion that said that each individual character is a pattern that matches itself? When you search for a word, you are already combining basic patterns.

You can combine any of the preceding grep patterns in the same way. Here are some examples.

Pattern	Matches	Examples
`\d+\+\d+`	a string of digits, followed by a literal plus sign, followed by more digits	4+2 1234+5829
`\d{4}[\t ]B\.C\.`	four digits, followed by a tab or a space, followed by the string B.C.	2152 B.C.
`\$?[0-9,]+\.\d*`	an optional dollar sign, followed by one or more digits and commas, followed by a period, then zero or more digits	1,234.56 $4,296,459.19 $3,5,6,4.00000. *(oops!)*

Note again in these examples how the characters that have special meaning to grep are preceded by a backslash (\+, \., and \$) when we want them to match themselves.

Creating Subpatterns

Subpatterns provide a means of organizing or grouping complex grep patterns. This is primarily important for two reasons: for limiting the scope of the alternation operator (which otherwise creates an alternation of everything to its left and right), and for changing the matched text when performing replacements.

A subpattern consists of any simple or complex pattern, enclosed in a pair of parentheses. You can optionally specify a simple string to identify a subpattern, making it a named subpattern.

Pattern	Matches
`(p)`	the pattern *p* and remembers it
`(?P<`*NAME*`>p)`	the pattern *p* and remembers it by the specified string *NAME*

You can combine more than one subpattern into a grep pattern, or mix subpatterns and other pattern elements as you need.

Taking the last set of examples, you could modify these to use subpatterns wherever actual data appears:

Pattern	Matches	Examples
`(\d+)\+(\d+)`	a string of digits, followed by a plus sign, followed by more digits	4+2 1234+5829
`(\d{4})[\t ]B\.C\.`	four digits, followed by a tab or a space, followed by the string B.C.	2152 B.C.
`\$?([0-9,]+)\.(\d*)`	an optional dollar sign, followed by one or more digits and commas, followed by a period, then zero or more digits	1,234.56 $4,296,459.19 $3,5,6,4.00000.

Using Backreferences in Subpatterns

What if we wanted to match a series of digits, followed by a plus sign, followed by the exact same series of digits as on the left side of the plus? In other words, we want to match "1234+1234" or "7+7", but *not* "5432+1984".

Using grouping parentheses, you can do this by referring to a backreference, also known as a captured subpattern. There are two kinds of backreferences: numbered backreferences, and named backreferences. You can use both types of backreference within the same grep pattern.

Each subpattern within the complete pattern is numbered from left to right, starting with the opening parenthesis. Later in the pattern, you can refer to the text matched within any of these subpatterns by using a backslash followed by the number of that subpattern; this is a numbered backreference. Unlike numbered backreferences, which are automatically identified from the pattern, named backreferences are only available after you define them.

Pattern	Matches…
`\1, \2, …, \99`	the text of the nth subpattern in the entire search pattern
`(?P=`*`NAME`*`)`	the text of the subpattern *NAME*

Names may include alphanumeric characters and underscores, and must be unique within a pattern.

Here are some examples of numbered backreferences:

Pattern	Matches	Examples
`(\d+)\+\1`	a string of digits, followed by a plus sign, followed the same digits	7+7 1234+1234
`(\w+)\s+\1`	double words, or, a pair of identical character runs separated by whitespace	the the ti**re re**turn *(oops!)*
`(\w)(\w)\2\1`	a word character, a second word character, followed by the second one again and the first one again	abba

We will revisit subpatterns in the section on replacement, where you will see how the choice of subpatterns affects the changes you can make.

Using Alternation

The alternation operator | allows you to match any of several patterns at a given point. To use this operator, place it between one or more patterns x|y to match either x or y.

As with all of the preceding options, you can combine alternation with other pattern elements to handle more complex searches.

Pattern	Text is…	Matches…
`a|t`	A cat	each "a" and "t"
`a|c|t`	A cat	each "a", "c", and "t"
`a (cat|dog) is`	A cat is here. A dog is here. A giraffe is here.	"A cat is", "A dog is"
`A|b+`	Abba	"A", "bb", and "a"
`Andy|Ted`	Andy and Ted joined AAA yesterday	"Andy" and "Ted"
`\d{4}|years`	I've been a loyal member since 1983, almost 16 years ago.	"1983", "years"
`[a-z]+|\d+`	That's almost 16 years.	"That", "s", "almost", "16", "years"

The "Longest Match" Issue

IMPORTANT When creating complex patterns, you should bear in mind that the quantifiers +, *, ? and {} are "greedy." That is, they will always make the longest possible match possible to a given pattern, so if your pattern is E+ (one or more E's) and your text contains "EEEE", the pattern matches all the E's at once, not just the first one. This is usually what you want, but not always.

Suppose, for instance, that you want to match an HTML tag. At first, you may think that a good way to do this would be to search for the pattern:

```
<.+>
```

consisting of a less-than sign, followed by one or more occurrences of a single character, followed by a greater-than sign. To understand why this may not work the way you think it should, consider the following sample text to be searched:

```
<B>This text is in boldface.</B>
```

The intent was to write a pattern that would match both of the HTML tags separately. Let's see what actually happens. The < character at the beginning of this line matches the beginning of the pattern. The next character in the pattern is . which matches any character (except a line break), modified with the + quantifier, taken together, this combination means one or more repetitions of any character. That, of course, takes care of the B. The problem is that the next > is also "any character" and that it *also* qualifies as "one or more repetitions." In fact, all of the text up to the end of the line qualifies as "one or more repetitions of any character" (the line break does not qualify, so grep stops there). After grep has reached the line break, it has exhausted the + operator, so it backs up and sees if it can find a match for >. Lo and behold, it can: the last character is a greater-than symbol. Success!

In other words, the pattern matches our entire sample line at once, *not the two separate HTML tags in it as we intended.* More generally, the pattern matches all the text in a given line or paragraph from the first < to the *last* >. The pattern only does what we intended when there is only one HTML tag in a line or paragraph. This is what we meant when we say that the regular quantifiers try to make the longest possible match.

Non-Greedy Quantifiers

IMPORTANT To work around this "longest match" behavior, you can modify your pattern to take advantage of *non-greedy* quantifiers.

Quantifier	Matches…
`+?`	one or more
`*?`	zero or more
`??`	zero or one
`{`*`COUNT`*`}?`	match exactly *COUNT* times
`{`*`MIN`*`,}?`	match at least *MIN* times
`{`*`MIN`*`,`*`MAX`*`}?`	match at least *MIN* times, but no more than *MAX*

Astute readers will note that these non-greedy quantifiers correspond exactly to their normal (greedy) counterparts, appended with a question mark.

Revisiting our problem of matching HTML tags, for example, we can search for:

```
<.+?>
```

This matches an opening bracket, followed by one or more occurrences of any character other than a return, followed by a closing bracket. The non-greedy quantifier achieves the results we want, preventing TextWrangler from "overrunning" the closing angle bracket and matching across several tags.

A slightly more complicated example: how could you write a pattern that matches all text between <B> and </B> HTML tags? Consider the sample text below:

```
<B>Welcome</B> to the home of <B>TextWrangler!</B>
```

As before, you might be tempted to write:

```
<B>.*</B>
```

but for the same reasons as before, this will match the entire line of text. The solution is similar; we will use the non-greedy *? quantifier:

```
<B>.*?</B>
```

Writing Replacement Patterns

Subpatterns Make Replacement Powerful

We covered subpatterns earlier when discussing search patterns and discussed how the parentheses can be used to limit the scope of the alternation operator. Another reason for employing subpatterns in your grep searches is to provide a powerful and flexible way to change or reuse found information as part of a search-and-replace operation. If you do not use subpatterns, you can still access the complete results of the search with the & metacharacter. However, this precludes reorganizing the matched data as it is replaced.

Pattern	Inserts…
`&`	the text matched by the entire search pattern
`\1, \2, …, \99`	the text matched by the nth subpattern of the entire search pattern
`\P<`*`NAME`*`>`	the text matched by the subpattern *NAME*

Note TextWrangler will remember up to 99 backreferenced subpatterns. Versions prior to 6.5 were limited to 9 subpatterns.

Using the Entire Matched Pattern

The & character is useful when you want to use the entire matched string as the basis of a replacement. Suppose that in your text every instance of product names that begin with the company name "ACME" needs to end with a trademark symbol (™). The following search pattern finds two-word combinations that begin with "ACME":

```
ACME [A-Za-z]+
```

The following replacement string adds the trademark symbol to the matched text:

```
&™
```

For example, if you start with

```
ACME Magnets, ACME Anvils, and ACME TNT are all premium
products.
```

and perform a replace operation with the above patterns, you will get:

```
ACME Magnets™, ACME Anvils™, and ACME TNT™ are all premium
products.
```

Using Parts of the Matched Pattern

While using the entire matched pattern in a replacement string is useful, it is often more useful to use only a portion of the matched pattern and to rearrange the parts in the replacement string.

For example, suppose a source file contains C-style declarations of this type:

```
#define Util_Menu 284
#define Tool_Menu 295
```

and you want to convert them so they look like this, Pascal-style:

```
const int Util_Menu = 284;
const int Tool_Menu = 295;
```

The pattern to find the original text is straightforward:

```
#define[ \t]+.+[ \t]+\d+[^0-9]*$
```

This pattern matches the word "#define" followed by one or more tabs or spaces, followed by one or more characters of any type, followed by one or more tabs or spaces, followed by one or more digits, followed by zero or more characters that are *not* digits (to allow for comments), followed by the end of the line.

The problem with this pattern is that it matches the entire line. It does not provide a way to remember the individual parts of the found string.

If you use subpatterns to rewrite the above search pattern slightly, you get this:

```
#define[ \t]+(.+)[ \t]+(\d+)[^0-9]*$
```

The first set of parentheses defines a subpattern which remembers the name of the constant. The second set remembers the value of the constant.

The replacement string would look like this:

```
const int \1 = \2;
```

The sequence \1 is replaced by the name of the constant (the first subpattern from the search pattern), and the sequence \2 is replaced by the value of the constant (from the second subpattern).

Our example throws out any comment that may follow the C-style constant declaration. As an exercise, try rewriting the search and replace patterns so they preserve the comment, enclosing it in (*…*) style Pascal comment markers.

Here are some more examples:

Data	Search for	Replace	Result
4+2	(\d+)\+(\d+)	\2+\1	2+4
1234+5829	(\d+)\+(\d+)	\1+\1	1234+1234
2152 B.C.	(\d{4})[\t]B\.C\.	\1 A.D.	2152 A.D.
1,234.56	\$?([0-9,]+)\.(\d+)	\1 dollars and \2 cents	1,234 dollars and 56 cents
$4,296,459.19	\$?([0-9,]+)\.(\d+)	\1 dollars and \2 cents	4,296,459 dollars and 19 cents
$3,5,6,4.00000	\$?([0-9,]+)\.(\d+)	\1 dollars and \2 cents	3,5,6,4 dollars and 00000 cents

Case Transformations

Replace patterns can also change the case of the original text when using subpattern replacements. The syntax is similar to Perl's, specifically:

Modifier	Effect
\u	Make the next character uppercase
\U	Make all following characters uppercase until reaching another case specifier (\u, \L, \l) or \E
\l	Make the next character lowercase
\L	Make all following characters lowercase until reaching another case specifier (\u, \U, \l) or \E
\E	End case transformation opened by \U or \L

Here are some examples to illustrate how case transformations can be used.

Given some text:

```
mumbo-jumbo
```

and the search pattern:

```
(\w+)(\W)(\w+)
```

the following replace patterns will produce the following output:

```
   \U\1\E\2\3    MUMBO-jumbo
\u\1\2\u\3       Mumbo-Jumbo
```

Note that case transformations also affect literal strings in the replace pattern:

```
   \U\1\2fred                MUMBO-FRED
\lMUMBLE\2\3              mUMBLE-jumbo
```

Finally, note that \E is not necessary to close off a modifier; if another modifier appears before an \E is encountered, that modifier will take effect immediately:

```
\Ufred-\uwilma              FRED-Wilma
```

Examples

The example patterns in this section describe some common character classes and shortcuts used for constructing grep patterns, and addresses some common tasks that you might find useful in your work.

Matching Identifiers

One of the most common things you will use grep patterns for is to find and modify identifiers, such as variables in computer source code or object names in HTML source documents. To match an arbitrary identifier in most programming languages, you might use this search pattern:

```
[a-z][a-zA-Z0-9]*
```

This pattern matches any sequence that begins with a lowercase letter and is followed by zero or more alphanumeric characters. If other characters are allowed in the identifier, add them to the pattern. This pattern allows underscores in only the first character of the identifier:

```
[a-z_][a-zA-Z0-9]*
```

The following pattern allows underscores anywhere *but* the first character, but allows identifiers to begin with an uppercase or lowercase letter:

```
[a-zA-Z][a-zA-Z0-9_]*
```

Matching White Space

Often you will want to match two sequences of data that are separated by tabs or spaces, whether to simply identify them, or to rearrange them.

For example, suppose you have a list of formatted label-data pairs like this:

```
User name:      Bernard Rubble
Occupation:     Actor
Spouse:         Betty
```

You can see that there are tabs or spaces between the labels on the left and the data on the right, but you have no way of knowing how many spaces or tabs there will be on any given line. Here is a character class that means “match one or more white space characters.”

```
[ \t]+
```

So, if you wanted to transform the list above to look like this:

```
User name("Bernard Rubble")
Occupation("Actor")
Spouse("Betty")
```

You would use this search pattern:

```
([a-z ]+):[ \t]+([a-z ]+)
```

and this replacement pattern:

```
\1\("\2"\)
```

Matching Delimited Strings

In some cases, you may want to match all the text that appears between a pair of delimiters. One way to do this is to bracket the search pattern with the delimiters, like this:

```
".*"
```

This works well if you have only one delimited string on the line. But suppose the line looked like this:

```
"apples", "oranges, kiwis, mangos", "penguins"
```

The search string above would match the entire line. (This is another instance of the "longest match" behavior of TextWrangler's grep engine, which was discussed previously.)

Once again, non-greedy quantifiers come to the rescue. The following pattern will match "-delimited strings:

```
".+?"
```

Marking Structured Text

Suppose you are reading a long text document that does not have a table of contents, but you notice that all the sections are numbered like this:

```
3.2.7     Prehistoric Cartoon Communities
5.19.001  Restaurants of the Mesozoic
```

You can use a grep pattern to create marks for these headings, which will appear in the Mark popup menu. Choose Find & Mark All from the Mark popup menu in the navigation bar. Then, decide how many levels you want to mark. In this example, the headings always have at least two digits and at most four.

Use this pattern to find the headings:

```
^(\d+\.\d+\.?\d*\.?\d*)[ \t]+([a-z ]+)
```

and this pattern to make the file marks:

```
\1 \2
```

The ^ before the first search group ensures that TextWrangler matches the numeric string at the beginning of a line. The pattern

```
\.?\d*
```

matches a (possible) decimal point and a digit sequence. The other groups use the white space idiom and the identifier idiom. You can use a similar technique to mark any section that has a section mark that can be described with grep.

Marking a Mail Digest

You can elaborate the structured text technique to create markers for mail digests. Assume that each digest is separated by the following lines:

```
From: Sadie Burke <sadie@burke.com>
Date: Sun, 16 Jul 1995 13:17:45 -0700
Subject: Fishing with the judge
```

Suppose you want the marker text to list the subject and the sender. You would use the following search string:

```
^From:[ \t]+(.*)\r.*\rSubject:[ \t]+(.*)
```

And mark the text with this replacement string:

```
\2 \1
```

Note that for the sequence \r.*\r in the middle of the search string, the \r before "Subject" is necessary because as previously discussed, the special character . does not match carriage returns. (At least, not by default. See "Advanced Topics," below, for details on how to make dot match *any* character, including carriage returns.)

Rearranging Name Lists

You can use grep patterns to transform a list of names in first name first form to last name first order (for a later sorting, for instance). Assume that the names are in the form:

```
Junior X. Potter
Jill Safai
Dylan Schuyler Goode
Walter Wang
```

If you use this search pattern:

```
^(.*) ([^ ]+)$
```

And this replacement string:

```
\2, \1
```

The transformed list becomes:

```
Potter, Junior X.
Safai, Jill
Goode, Dylan Schuyler
Wang, Walter
```

Advanced Grep Topics

TextWrangler's PCRE-based grep engine offers unparalleled syntactical power. The topics below cover areas that show how grep can effectively match very complicated patterns of textTextWrangler. However, with this power comes complexity.

If you are new to grep, it is possible that the topics covered in this section will not make much sense to you. That's OK. The best way to learn grep is to use it in real life, not by reading example patterns. In many cases, the basic grep syntax covered previously in this chapter will be all that you need.

If you are an experienced user of grep, however, many of the topics covered below will be of great interest.

Matching Nulls

The grep engine used in much older versions of TextWrangler (prior to 2.0) was unable to search text that contained null characters (ASCII value zero), but this limitation has since been removed. Here's one way to match a null:

```
\x{0}
```

Backreferences

The following charts explain the rules TextWrangler uses for determining backreferences.

In Search Patterns

Modifier	Effect
\0	A backslash followed by a zero is an octal character reference. Up to two further octal characters are read. Thus, "\040" will match a space character, and "\07" will match the ASCII BEL (\x07), but "\08" will match an ASCII null followed by the digit 8 (because octal characters only range from 0-7).
\1-9	A backslash followed by a single decimal digit from 1 to 9 is always a backreference to the *Nth* captured subpattern.
\10-99	A backslash followed by two decimal digits, which taken together form the integer N (ranging from 10 to 99), is a backreference to the *Nth* captured subpattern, *if* there exist *N* capturing sets of parentheses in the pattern. If there are fewer than *N* captured subpatterns, the grep engine will instead look for up to three octal digits following the backslash. Any subsequent digits stand for themselves. So, in a search pattern, "\11" is a backreference if there are 11 or more sets of capturing parentheses in the pattern. If not, it matches a tab. "\011" always matches a tab. "\81" is a backreference if there are 81 or more captured subpatterns, but matches an ASCII null followed by the two characters "8" and "1" otherwise.

In Character Classes

Modifier	Effect
OCTAL	Inside a character class, a backslash followed by up to three octal digits generates a single byte character reference from the least significant eight bits of the value. Thus, the character class "[\7]" will match a single byte with octal value 7 (equivalent to "\x07"). "[\8]" will match a literal "8" character.

In Replacement Patterns

Modifier	Effect
NNN+	If more than two decimal digits follow the backslash, only the first two are considered part of the backreference. Thus, "\111" would be interpreted as the 11th backreference, followed by a literal "1". You may use a leading zero; for example, if in your replacement pattern you want the first backreference followed by a literal "1", you can use "\011". (If you use "\11", you will get the 11th backreference, even if it is empty.)
NN	If two decimal digits follow the backslash, which taken together represent the value N, and if there is an Nth captured substring, then all three characters are replaced with that substring. If there is not an Nth captured substring, all three characters are discarded—that is, the backreference is replaced with the empty string.
N	If there is only a single digit N following the backslash and there is an Nth captured substring, both characters are replaced with that substring. Otherwise, both characters are discarded—that is, the backreference is replaced with the empty string. In replacement patterns, \0 is a backreference to the entire match (exactly equivalent to "&").

POSIX-Style Character Classes

TextWrangler now provides support for POSIX-style character classes. These classes are used in the form [:CLASS:], and are *only* available inside regular character classes (in other words, inside another set of square brackets).

Class	Meaning
alnum	letters and digits
alpha	letters
ascii	character codes 0-127
blank	horizontal whitespace
cntrl	control characters
digit	decimal digits (same as \d)
graph	printing characters, excluding spaces

Class	Meaning
lower	lower case letters
print	printing characters, including spaces
punct	punctuation characters
space	white space (same as \s)
upper	upper case letters
word	"word" characters (same as \w)
xdigit	hexadecimal digits

For example: [[:digit:]]+ is the same as: [\d]+

POSIX-style character class names are case-sensitive.

It is easy to forget that POSIX-style character classes are only available inside regular character classes. The pattern [:space:], without enclosing square brackets, is just a character class consisting of the characters ":", "a", "c", "e", "p", and "s".

The names "ascii" and "word" are Perl extensions; the others are defined by the POSIX standard. Another Perl extension supported by TextWrangler is negated POSIX-style character classes, which are indicated by a ^ after the colon. For example, to match any run of non-digit characters:

```
[[:^digit:]]+
```

Non-Capturing Parentheses

As described in the preceding section "Creating Subpatterns", bare parentheses cluster and capture the subpatterns they contain. The portion of the matching pattern contained within the first pair of parentheses is available in the backreference \1, the second in \2, and so on.

Opening parentheses are counted from left to right to determine the numbers of the captured subpatterns. For example, if the following grep pattern:

```
((red|white) (king|queen))
```

is matched against the text "red king", the backreferences will be set as follows:

```
\1  "red king"
\2  "red"
\3  "king"
```

Sometimes, however, parentheses are needed only for clustering, not capturing. TextWrangler now supports non-capturing parentheses, using the syntax:

```
(?:PATTERN)
```

That is, if an open parenthesis is followed by "?:", the subpattern matched by that pair of parentheses is not counted when computing the backreferences. For example, if the text "red king" is matched against the pattern:

```
(?:(red|white) (king|queen))
```

the backreferences will be set as follows:

```
\1  "red"
\2  "king"
```

Perl-Style Pattern Extensions

TextWrangler's grep engine supports several extended sequences, which provide grep patterns with super-powers from another universe. Their syntax is in the form:

```
(?KEY…)
```

in other words, an open parenthesis followed by a question mark, followed by a KEY for the particular grep extension, followed by the rest of the subpattern and a closing parenthesis.

We have already seen one such extension in the previous section of this document—non-capturing parentheses: (?:…). The remainder are listed in the chart below, and discussed in detail afterward.

Extension	Meaning
(?:…)	Cluster-only parentheses, no capturing
(?#…)	Comment, discard all text between the parentheses
(?imsx-imsx)	Enable/disable pattern modifiers
(?imsx-imsx:…)	Cluster-only parens with modifiers
(?=…)	Positive lookahead assertion
(?!…)	Negative lookahead assertion
(?<=…)	Positive lookbehind assertion
(?<!…)	Negative lookbehind assertion
(?()…|…)	Match with if-then-else
(?()…)	Match with if-then
(?>…)	Match non-backtracking subpattern ("once-only")
(?R)	Recursive pattern

Comments

The sequence (?# marks the start of a comment which continues up to the next closing parenthesis. Nested parentheses are not permitted. The characters that make up a comment play no part in the pattern matching at all.

Search for: `foo(?# Hello, this is a comment)bar`
Will match: foobar

Pattern Modifiers

The settings for case sensitivity, multi-line matching, whether the dot character can match returns, and "extended syntax" can be turned on and off within a pattern by including sequences of letters between "(?" and ")".

Modifier	Meaning	Default
`i`	case insensitive	according to Case Sensitive checkbox in Find window
`m`	allow ^ and $ to match at \r	on
`s`	allow . to match \r	off
`x`	ignore most white space and allow inline comments in grep patterns	off

i — By default, TextWrangler obeys the "Case Sensitive" checkbox in the Find window (or the corresponding property of the search options when using the scripting interface). The (?i) option overrides this setting.

m — By default, TextWrangler's grep engine will match the ^ and $ metacharacters after and before returns, respectively. If you turn this option off with (?-m), ^ will only match at the beginning of the document, and $ will only match at the end of the document. (If that is what you want, however, you should consider using the new \A, \Z, and \z metacharacters instead of ^ and $.)

s —By default, the magic dot metacharacter . matches any character except return ("\r"). If you turn this option on with (?s), however, dot will match *any* character. Thus, the pattern (?s).+ will match an entire document.

x —When turned on, this option changes the meaning of most whitespace characters (notably, tabs and spaces) and #. Literal whitespace characters are ignored, and the # character starts a comment that extends until a literal return or the "\r" escape sequence is encountered. Ostensibly, this option intends to let you write more "readable" patterns.

Perl programmers should already be familiar with these options, as they correspond directly to the -imsx options for Perl's m// and s/// operators. Unadorned, these options turn their corresponding behavior on; when preceded by a hyphen (-), they turn the behavior off. Setting and unsetting options can occur in the same set of parentheses.

Example	Effect
`(?imsx)`	Turn all four options on
`(?-imsx)`	Turn all four options off
`(?i-msx)`	Turn "i" on, turn "m", "s", and "x" off

The scope of these option changes depends on where in the pattern the setting occurs. For settings that are outside any subpattern, the effect is the same as if the options were set or unset at the start of matching. The following patterns all behave in exactly the same way:

```
(?i)abc
a(?i)bc
ab(?i)c
abc(?i)
```

In other words, all four of the above patterns will match without regard to case. Such "top level" settings apply to the whole pattern (unless there are other changes inside subpatterns). If there is more than one setting of the same option at the top level, the right-most setting is used.

If an option change occurs inside a subpattern, the effect is different. An option change inside a subpattern affects *only* that part of the subpattern that follows it, so, if the "Case Sensitive" checkbox is turned on:

Search for: `(a(?i)b)c`
Will match: `abc` *or* `aBc`

and will not match anything else. (But if "Case Sensitive" is turned off, the "(?i)" in the above pattern is superfluous and has no effect.) By this means, options can be made to have different settings in different parts of the pattern. Any changes made in one alternative do carry on into subsequent branches within the same subpattern. For example:

Search for: `(a(?i)b|c)`

matches "ab", "aB", "c", and "C", even though when matching "C", the first branch is abandoned before the option setting.

These options can also be set using the clustering (non-capturing) parentheses syntax defined earlier, by inserting the option letters between the "?" and ":". The scope of options set in this manner is limited to the subpattern contained therein. Examples:

Search for: `(?i:saturday|sunday)`
Will match: `SATURDAY` *or* `Saturday` *or* `SUNday` (and so on)

Search for: `(?i:foo)(?-i:bar)`
Will match: `foobar` *or* `FOObar`
Will not match: `FOOBAR` *or* `fooBAR`

Positional Assertions

Positional assertions "anchor" a pattern, without actually matching any characters. Simple assertions have already been described: those which are invoked with the escape sequences \b, \B, \A, \Z, \z, ^ and $. For example, the pattern \bfoo\b will only match the string "foo" if it has word breaks on both sides, but the \b's do not themselves match any characters; the entire text matched by this pattern are the three characters "f", "o", and "o".

Lookahead and lookbehind assertions work in a similar manner, but allow you to test for arbitrary patterns to anchor next to. If you have ever said to yourself, "I would like to match 'foo', but only when it is next to 'bar'," lookaround assertions fill that need.

Positive lookahead assertions begin with "(?=", and negative lookahead assertions begin with "(?!". For example:

```
\w+(?=;)
```

will match any word followed by a semicolon, but the semicolon is not included as part of the match.

```
foo(?!bar)
```

matches any occurrence of "foo" that is *not* followed by "bar". Note that the apparently similar pattern:

```
(?!foo)bar
```

does not find an occurrence of "bar" that is preceded by something other than "foo"; it finds any occurrence of "bar" whatsoever, because the assertion (?!foo) is always true when the next three characters are "bar". A lookbehind assertion is needed to achieve this effect.

Positive lookbehind assertions start with "(?<=", and negative lookbehind assertions start with "(?<!". For example:

```
(?<!foo)bar
```

does find an occurrence of "bar" that is not preceded by "foo". The contents of a lookbehind assertion are restricted such that all the strings it matches must have a fixed length. However, if there are several alternatives, they do not all have to have the same fixed length. Thus

```
(?<=Martin|Lewis)
```

is permitted, but

```
(?<!dogs?|cats?)
```

causes an error. Branches that match different length strings are permitted only at the top level of a lookbehind assertion. This is different compared with Perl 5.005, which requires all branches to match the same length of string. An assertion such as

```
(?<=ab(c|de))
```

is not permitted, because its single top-level branch can match two different lengths, but it is acceptable if rewritten to use two top-level branches:

```
(?<=abc|abde)
```

The implementation of lookbehind assertions is, for each alternative, to temporarily move the current position back by the fixed width and then try to match. If there are insufficient characters before the current position, the match is deemed to fail. (Lookbehinds in conjunction with non-backtracking [a.k.a. "once-only"] subpatterns can be particularly useful for matching at the ends of strings; an example is given in the section on once-only subpatterns below.)

Several assertions (of any sort) may occur in succession. For example,

```
(?<=\d{3})(?<!999)foo
```

matches “foo” preceded by three digits that are not “999”. Notice that each of the assertions is applied independently at the same point in the subject string. First there is a check that the previous three characters are all digits, and then there is a check that the same three characters are not “999”. This pattern does not match “foo” preceded by six characters, the first of which are digits and the last three of which are not “999”. For example, it does not match “123abcfoo”. A pattern to do that is:

```
(?<=\d{3}...)(?<!999)foo
```

This time the first assertion looks at the preceding six characters, checking that the first three are digits, and then the second assertion checks that the preceding three characters are not “999”. Assertions can be nested in any combination. For example,

```
(?<=(?<!foo)bar)baz
```

matches an occurrence of “baz” that is preceded by “bar” which in turn is not preceded by “foo”, while

```
(?<=\d{3}(?!999)...)foo
```

is another pattern which matches “foo” preceded by three digits and any three characters that are not “999”.

Assertion subpatterns are not capturing subpatterns, and may not be repeated, because it makes no sense to assert the same thing several times. If any kind of assertion contains capturing subpatterns within it, these are counted for the purposes of numbering the capturing subpatterns in the whole pattern. However, substring capturing is carried out only for positive assertions, because it does not make sense for negative assertions.

Conditional Subpatterns

Conditional subpatterns allow you to apply “if-then” or “if-then-else” logic to pattern matching. The “if” portion can either be an integer between 1 and 99, or an assertion.

The forms of syntax for an ordinary conditional subpattern are:

if-then:	`(?(condition)yes-pattern)`
if-then-else:	`(?(condition)yes-pattern|no-pattern)`

and for a named conditional subpattern are:

if-then:	`(?P<`*NAME*`>(condition)yes-pattern)`
if-then-else:	`(?P<`*NAME*`>(condition)yes-pattern|no-pattern)`

If the condition evaluates as true, the “yes-pattern” portion attempts to match. Otherwise, the “no-pattern” portion does (if there is a “no-pattern”).

If the “condition” text between the parentheses is an integer, it corresponds to the backreferenced subpattern with the same number. (Do not precede the number with a backslash.) If the corresponding backreference has previously matched in the pattern, the condition is satisfied. Here’s an example of how this can be used. Let’s say we want to match the words “red” or “blue”, and refer to whichever word is matched in the replacement pattern. That’s easy:

```
(red|blue)
```

To make it harder, let's say that if (and only if) we match "blue", we want to optionally match a space and the word "car" if they follow directly afterward. In other words, we want to match "red", "blue", or if possible, "blue car", but we do not want to match "red car". We cannot use the pattern:

```
(red|blue)( car)?
```

because that will match "red car". Nor can we use:

```
(red|blue car|blue)
```

because in our replacement pattern, we want the backreference to only contain "red" or "blue", without the " car". Using a conditional subpattern, however, we can search for:

```
((blue)|(red))(?(2) car)?
```

Here's how this pattern works. First, we start with "((blue)|(red))". When this subpattern matches "blue", \1 and \2 are set to "blue", and \3 is empty. When it matches "red", \1 and \3 are set to "red", and \2 is empty.

Next comes the conditional subpattern "(?(2) car)?". The conditional test is on "2", the second backreferenced subpattern: if \2 is set, which in our case means it has matched the word "blue", then it will try to match " car". If \2 is not set, however, the entire conditional subpattern is skipped. The question mark at the end of the pattern makes this conditional match optional, even if \2 is set to "blue".

Here's an example that uses an assertion for the condition, and the if-then-else form. Let's say we want to match a run of digits of any length, followed by either " is odd" or " is even", depending on whether the matched digits end with an odd or even digit.

```
\d+(?(?<=[13579]) is odd| is even)
```

This pattern starts with "\d+" to match the digits. Next comes a conditional subpattern, with a positive lookbehind assertion as the condition to be satisfied. The lookbehind assertion is true only if the last character matched by \d+ was also in the character class [13579]. If that is true, we next try to match " is odd"; if it is not, we try to match " is even". Thus, this pattern will match "123 is odd", "8 is even", and so on, but will not match "9 is even" or "144 is odd".

Once-Only Subpatterns

With both maximizing (greedy) and minimizing (non-greedy) repetition, failure of what follows normally causes the repeated item to be reevaluated to see if a different number of repeats allows the rest of the pattern to match. Sometimes it is useful to prevent this, either to change the nature of the match, or to cause it to fail earlier than it otherwise might, when the author of the pattern knows there is no point in carrying on.

Consider, for example, the pattern "\d+foo" when matching against the text "123456bar".

After matching all 6 digits and then failing to match “foo”, the normal action of the grep engine is to try again with only 5 digits matching the \d+ item, and then with 4, and so on, before ultimately failing. Once-only subpatterns provide the means for specifying that once a portion of the pattern has matched, it is not to be reevaluated in this way, so the matcher would give up immediately on failing to match “foo” the first time. The notation is another kind of special parenthesis, starting with “(?>”, as in this example:

```
(?>\d+)bar
```

This kind of parentheses “locks up” the part of the pattern it contains once it has matched, and a failure further into the pattern is prevented from backtracking into it. Backtracking past it to previous items, however, works as normal.

In most situations, such as in the example above, the time saved by using once-only subpatterns is insignificant—a few small fractions of a second, at most. With some complicated grep patterns or with humongous lines of text, however, you can save tremendous amounts of time using once-only subpatterns.

Once-only subpatterns are not capturing subpatterns. Simple cases such as the above example can be thought of as a maximizing repeat that must swallow everything it can. So, while both \d+ and \d+? are prepared to adjust the number of digits they match in order to make the rest of the pattern match, (?>\d+) can only match an entire sequence of digits.

Once-only subpatterns can be used in conjunction with lookbehind assertions to specify efficient matching at the end of a line of text. Consider a simple pattern such as:

```
abcd$
```

when applied to a long line of text which does not match (in other words, a long line of text that does not end with “abcd”). Because matching proceeds from left to right, the grep engine will look for each “a” in the subject and then see if what follows matches the rest of the pattern. If the pattern is specified as:

```
^.*abcd$
```

the initial .* matches the entire line at first, but when this fails (because there is no following “a”), it backtracks to match all but the last character, then all but the last two characters, and so on. Once again the search for “a” covers the entire string, from right to left, so we are no better off. However, if the pattern is written as:

```
^(?>.*)(?<=abcd)
```

there can be no backtracking for the .* item; it can match only the entire line. The subsequent lookbehind assertion does a single test on the last four characters. If it fails, the whole match fails immediately. For long strings, this approach makes a significant difference to the processing time.

When a pattern contains an unlimited repeat inside a subpattern that can itself be repeated an unlimited number of times, the use of a once-only subpattern is the only way to avoid some failing matches taking a very long time (literally millions or even billions of years, in some cases!). The pattern:

```
(\D+|<\d+>)*[!?]
```

matches an unlimited number of substrings that either consist of non-digits, or digits enclosed in <>, followed by either ! or ?. When it matches, it runs quickly. However, if it is attempts to match this line of text:

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

it takes a long time before reporting failure. So long, in fact, that it will effectively "freeze" TextWrangler. This is not really a crash, per se, but left to run on its own, it might take years before it finally fails. (We are not sure, frankly, because much like determining how many licks it takes to get to the center of a Tootsie Pop, we do not feel like waiting long enough to find out.)

The reason this takes so long to fail is because the string can be divided between the two repeats in a large number of ways, and all have to be tried before the grep engine knows for certain that the pattern will not match. (The example used [!?] rather than a single character at the end, because both PCRE and Perl have an optimization that allows for fast failure when a single character is used. They remember the last single character that is required for a match, and fail early if it is not present in the string.) If the pattern is changed to

```
((?>\D+)|<\d+>)*[!?]
```

sequences of non-digits cannot be broken, and failure happens quickly.

Recursive Patterns

Consider the problem of matching a string in parentheses, allowing for unlimited nested, balanced parentheses. Without the use of recursion, the best that can be done is to use a pattern that matches up to some fixed depth of nesting. It is not possible to handle an arbitrary nesting depth. Perl 5.6 has provided an experimental facility that allows regular expressions to recurse (among other things). It does this by interpolating Perl code in the expression at run time, and the code can refer to the expression itself. Obviously, TextWrangler's grep engine cannot support the interpolation of Perl code. Instead, the special item (?R) is provided for the specific case of recursion. The following recursive pattern solves the parentheses problem:

```
\(((?>[^()]+)|(?R))*\)
```

First it matches an opening parenthesis. Then it matches any number of substrings which can either be a sequence of non-parentheses, or a recursive match of the pattern itself (that is, a correctly parenthesized substring). Finally there is a closing parenthesis.

This particular example pattern contains nested unlimited repeats, and so the use of a once-only subpattern for matching strings of non-parentheses is important when applying the pattern to strings that do not match. For example, when it tries to match against this line of text:

```
(aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa()
```

it yields "no match" quickly. However, if a once-only subpattern is not used, the match runs for a very long time indeed because there are so many different ways the + and * repeats can carve up the subject, and all have to be tested before failure can be reported.

Recommended Books and Resources

Mastering Regular Expressions, 3rd Edition

by Jeffrey E.F. Friedl. O'Reilly & Associates, 2006. ISBN 0-596-52812-4

Although it does not cover TextWrangler's grep features specifically, *Mastering Regular Expressions* is an outstanding resource for learning the "how-to" of writing useful grep patterns, and the second edition is even better than the original.

TextWrangler Talk

The TextWrangler Talk discussion group covers a wide range of topics and questions about using TextWrangler, which frequently include searching and the use of grep patterns.

```
http://groups.google.com/group/textwrangler
```

Note TextWrangler's grep engine is based on the PCRE library package, which is open source software, written by Philip Hazel, and copyright 1997-2004 by the University of Cambridge, England. For details, see: http://www.pcre.org/

CHAPTER

9 Browsers

Browsers are special kinds of windows that let you see a lot of information about files at once. Browsers typically have two panes: one pane lets you select a file, the other displays detailed information about the file (often its contents). If you have performed a Find All search, you have already seen an example of a TextWrangler browser.

In this chapter

Browser Overview **169**
List Pane – 169 • *Toolbar* – 170
Text View Pane – 170 • *Splitter* – 170
Disk Browsers **171**
Disk Browser Controls – 171
Contextual Menu Commands – 172
Dragging Items – 172
Using the List Pane in Disk Browsers – 172
Search Results Browsers **173**
Error Results Browsers **174**

Browser Overview

All TextWrangler browsers share the same basic structure and behavior. All browsers have a toolbar, a file list, and a text pane. You can either edit files directly in any browser window, or open them separately.

List Pane

The top pane of a browser lists the items available in the browser. This pane shows different information for different kinds of browsers:

Browser	File List pane contains
Disk browser	Files and folders that TextWrangler can open
Search results	File and line number of each match
Error results (or) general results	File, line number, and status message for each condition

You can open both files and folders from the list pane. When you double-click a folder name, TextWrangler replaces the file list pane with the contents of the folder. When you double-click a file name, TextWrangler opens the file in an editing window. If the file list pane also included a line number, TextWrangler scrolls to that line.

Controls above the list may allow you to determine what kinds of items are displayed in the list. For example, in disk browsers, there is a popup menu that lets you choose to display text files, all files, or other types of files, and another that lets you return the browser to a parent directory of the current folder. In error browsers, checkboxes allow you to hide or show all errors, warnings, or notes.

For results browsers, TextWrangler shows a hierarchical listing, where all the results associated with a particular file are grouped under that file, using disclosure triangles similar to those in the Finder's list views to reveal or hide the results list.

To remove items from the display list, select them and press the Delete key, or choose Clear from the Edit menu.

In results browsers, you may Control-click on items in the list to bring up the contextual menu with relevant commands, such as "Copy".

Toolbar

The browser toolbar is like the toolbar in editing windows. Some browsers have additional buttons and controls in the status area as well.

These standard items—the pencil icon; the Function, Text Options, Mark, Path popup menus; and the Info buttons—should already be familiar to you, since they appear on TextWrangler document windows by default. See "Window Anatomy" in Chapter 4 for an explanation of these standard TextWrangler functions.

Text View Pane

When you click on a file name in the list pane, TextWrangler displays that file in the text view pane, and you can edit the file just as if it were open in a document window.

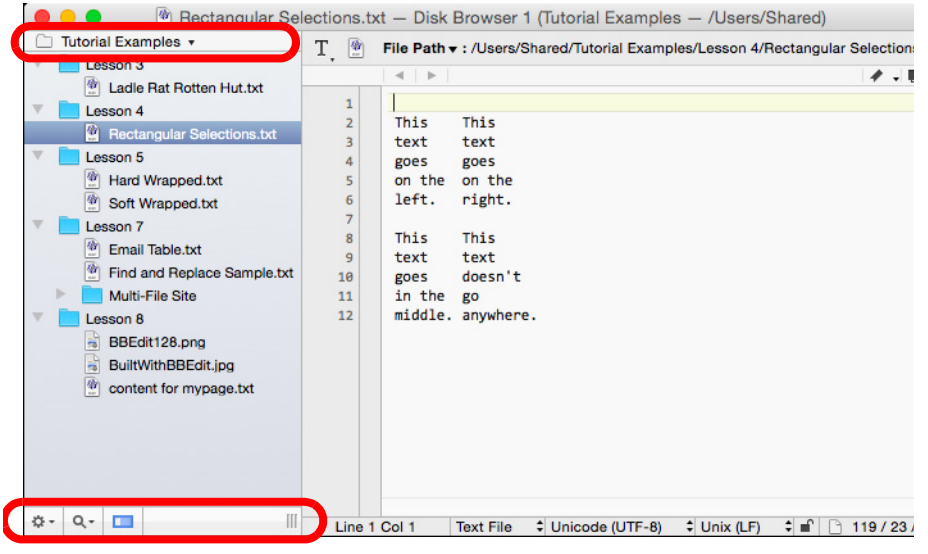
Splitter

You can change the size of the file list pane or the text view pane by dragging the double line that separates them. Double-clicking on the splitter bar will collapse the text view pane completely, and double-clicking on it again (in the bottom of the browser window) will restore the text pane to its previous proportions. You can also choose the Hide Editor or View Editor commands in the View menu to hide or display the text view pane.

Disk Browsers

Use a disk browser to explore the contents of a disk or a folder without opening each file one at a time.

To open a disk browser, pull down the File menu and choose Disk Browser from the New submenu. TextWrangler opens a new disk browser that starts in your home directory, but you can navigate to any desired location:



The name and path of the file (if any) and directory currently being viewed are displayed in the title bar of the window. The file list pane displays all the items in the current folder. Click on a file in the file list pane to open it in the text pane, or double-click to open the file into a text window.

Disk Browser Controls

The menus at the top and bottom of the file list pane let you create new files and folders, open existing files and folders, reveal them in the Finder or navigate to them in the Terminal, limit the kinds of files to show in the list pane, and navigate through your disks and folders.

Directory Menu

The Directory popup menu at the top of the file list pane always shows the currently active folder. You can use this menu to "back out" of any folder you are currently in to a higher-level folder (as you can by Command-clicking the name of a folder in the Finder).

Action Menu

The commands on the Action (gear) popup menu at the bottom of the file list pane allow you to open the selected items, reveal them in the Finder, copy their paths, navigate to their location in the Terminal, move them to the Trash, or create a new file or folder.

Filter Menu

The Filter (magnifying glass) popup menu at the bottom of the file list pane lets you specify what kinds of files TextWrangler should display:

- All Available: All files which TextWrangler recognizes, including its own document types. This includes text files, images, text factories, and so on.
- Text Files Only: Only files which TextWrangler recognizes as text files.
- Everything: All items present, including invisible files and folders.

You can also select a file filter to further limit what files TextWrangler should display. (You can define additional file filters in the Filters panel of the Setup window.)

Toggle Editor Button

Click this button to collapse or expand the browser's text view pane. (This button has the same effect as choosing the View/Hide Editor command in the View menu.)

Contextual Menu Commands

If you select one or more items in the file list pane and bring up the contextual menu, TextWrangler will offer a variety of commands including those available from the Action menu.

Dragging Items

You can select and drag files and folders from a disk browser's file list to any location, either within TextWrangler or elsewhere, which can accept file or folder drags. For example, you can drag a file from a disk browser to an editing window to insert its contents, or to a folder in the Finder to copy or move it.

Using the List Pane in Disk Browsers

The list pane of a disk browser displays disks, files, and folders. When you are at the computer level, the list shows all mounted volumes.

When you click a folder or disk in the list pane, TextWrangler displays the names of all the files it can open in the text pane, subject to the criteria specified by the Show and Filter menus.

When you click a file name in the list pane, TextWrangler displays that file in the text pane.

To open a folder or disk and display its contents in the file list pane, you can either double-click it, or Select it and press Command-Down Arrow.

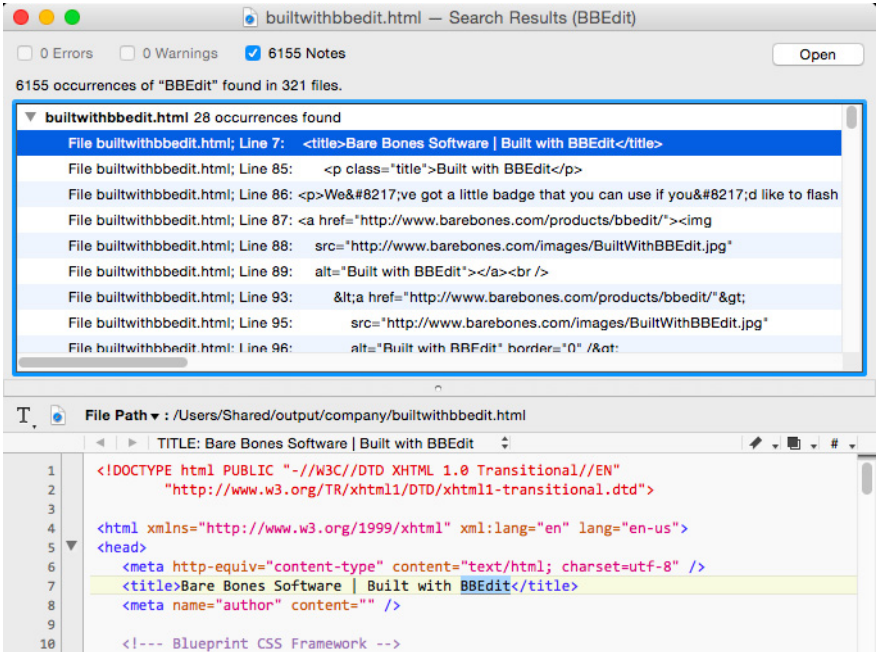
To go up one level to the enclosing folder or disk, you can either choose the enclosing folder from the directory popup menu, or press Command-Up Arrow

You can also use Quick Look to examine any non-text file by selecting it and pressing the spacebar.

Note When the list pane has input focus, the browser window's AppleScript "selection" property will return a list of the files currently selected. See "Getting and Setting Properties" on page 218 for further details.

Search Results Browsers

If you selected the Batch Find option when performing a multi-file search, TextWrangler displays every occurrence of the search string in the searched files in a search results browser.



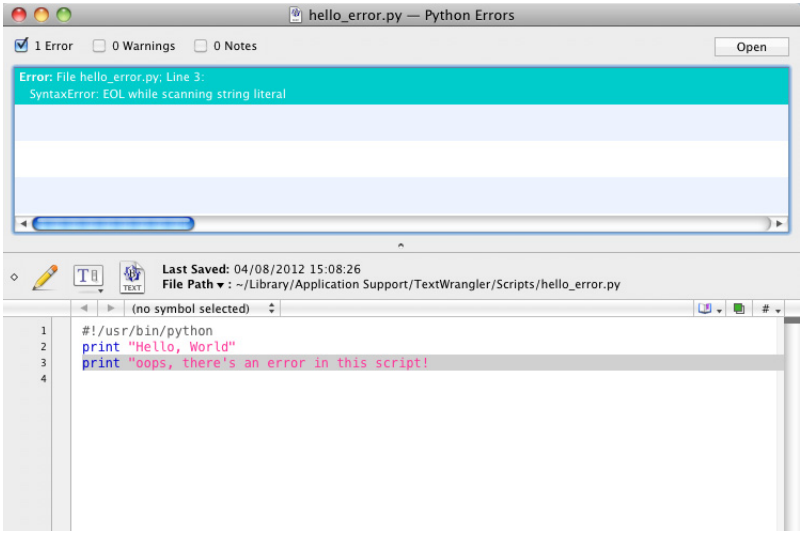
The items at the top of the window tell you how many matches TextWrangler found in the set of files you specified, as well as whether any error conditions or warnings were generated during the search. The list pane lists each line that contains the matched text. Every match is identified by file and line number. To choose whether to display the search errors, warnings, and results, use the checkboxes at the top of the browser.

To open the file which contains a particular match, just click on that match in the results list. After you have opened a file, you can use the Search menu commands to continue searching it. (See Chapter 7 for more information on searching.)

The Open button opens the selected items using TextWrangler. To open the selected items using the Finder, hold down the Option key while clicking the Open button.

Error Results Browsers

When you check the syntax of a Unix script, or run a script which generates any errors, TextWrangler will open an error results browser to list those errors.



Each entry in the list pane corresponds to an error, warning, or note. You can use the checkboxes for each type of item to suppress or display the associated results as desired.

If you click on a entry in the file list, TextWrangler will open the corresponding file in the text display pane and select the section of text related to the error.

CHAPTER

10 Preferences

You can use the Preferences command to customize much of TextWrangler's behavior. You can decide which windows are open when you launch TextWrangler, set the default options for windows, set the default options for searches, and so on. This chapter describes TextWrangler's extensive preference options.

In this chapter

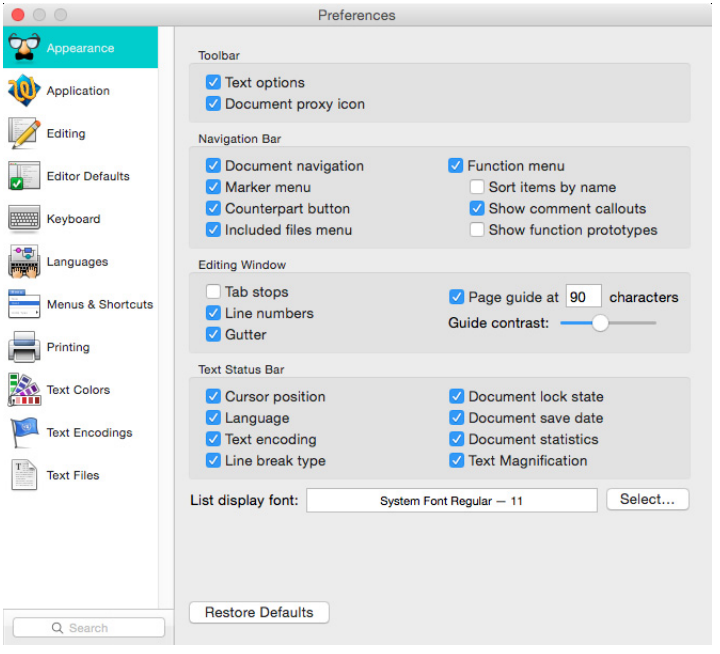
The Preferences Window **175**
- *Searching the Preferences* – 177
- *Appearance Preferences* – 177
- *Application Preferences* – 180
- *Editing Preferences* – 182
- *Editor Defaults Preferences* – 183
- *Keyboard Preferences* – 185
- *Languages Preferences* – 186
- *Menus & Shortcuts Preferences* – 188
- *Printing Preferences* – 189
- *Text Colors Preferences* – 190
- *Text Encodings Preferences* – 191
- *Text Files Preferences* – 192
- *Expert preferences settings* – 194

The Setup Window **194**
- *Bookmarks* – 194
- *Filters* – 194
- *Patterns* – 195

The Preferences Window

The Preferences window provides control over many aspects of TextWrangler's behavior. You can decide which actions TextWrangler should perform when you launch it, set default options for editing behavior, examine and set or modify keyboard shortcuts, create and apply text color schemes, and so on.

To open the Preferences window, choose the Preferences command from the TextWrangler (application) menu.



To select a preference panel, click its name in the list at the left side of the window. The text area at the top of the Preferences window gives you a brief description of the options provided by the currently displayed preference panel.

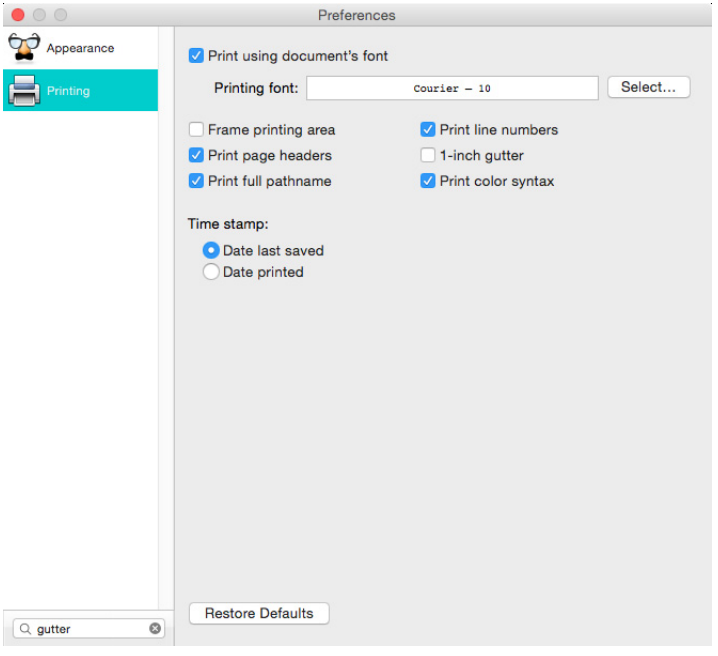
TextWrangler's Preferences window is non-modal: you can leave it open and change preference settings while you work, or close it at any time by clicking its close button or by choosing Close Window from the File menu. Any changes you make to preference options take effect immediately unless otherwise indicated.

IMPORTANT TextWrangler employs the standard system preferences mechanism to store your preference settings. Accordingly, you can modify preference options directly by issuing "defaults write" commands. However, if you choose to modify your preferences by means of "defaults write" commands other than those documented in this manual or the "Expert Preferences" page of TextWrangler's built-in Help book, without explicit advice from Bare Bones Software technical support, you take responsibility for any adverse effects.

If you discard your TextWrangler preferences file, you will need to re-select any customized preference options you may have chosen.

Searching the Preferences

You can perform keyword searches to quickly locate preference options in the Preferences window. To do this, just click in the search field below the list of preference panels, and type a word or partial word into the field. As you type, TextWrangler will search for instances of the current term and display all the panels which contain it. You can then select any of the listed panels to view and change the options within it. For example, here is the Preferences window with an active search for the term “gutter”.



Restore Defaults

Each of TextWrangler’s preference panels contains this button, which you can click to reset all preference options within the current panel to their factory default settings.

Appearance Preferences

The Appearance preferences let you choose which control and display elements appear in text windows and in other windows which include text panes.

Toolbar

When any of the listed options are on, TextWrangler displays the toolbar (see page 63). You can also show or hide the toolbar independently in any editing window.

Text options

When this option is on, TextWrangler displays the Text Options popup in the toolbar (see page 63).

Document proxy icon
When this option is on, TextWrangler displays the document proxy icon in the toolbar (see page 63). This icon serves as a proxy for the document file; you can click it to reveal the current file in the Finder, or drag it anywhere the original file can be dragged.

Navigation Bar
When any of the listed options are on, TextWrangler displays the navigation bar (see page 65). You can also show or hide the navigation bar independently for each text window. This option is on by default.

Document Navigation
When this option is on, TextWrangler displays the Previous and Next buttons and the Document popup menu in the navigation bar (see page 65).

Marker menu
When this option is on, TextWrangler displays the Marker popup menu in the navigation bar (see page 65).

Counterpart button
When this option is on, TextWrangler displays the Counterpart button in the navigation bar (see page 65).

Included files menu
When this option is on, TextWrangler displays the Included Files popup menu in the navigation bar (see page 65).

Function menu
When this option is on, TextWrangler displays the Function popup menu in the navigation bar (see page 65). The related options below control how items appear in the menu.

Sort items by name
If this option is on, TextWrangler sorts the items in the Function popup menu by name. Otherwise, items appear in the same order in the menu as they appear in the file. This option is off by default.

Show comment callouts
When this option is on, TextWrangler will suppress callouts embedded in comments from appearing in the Function popup. This option is on by default.

Show function prototypes
When this option is on, TextWrangler displays the names of function prototypes as well as function definitions in the Function popup menu. Otherwise, the menu does not include entries for function prototypes. This option is on by default.

Editing Window
These options control additional elements which TextWrangler can display in editing windows.

Tab stops
If this option is on, TextWrangler displays tab stops as vertical grid lines within the content area of text windows, using the tab width set in the Editor Defaults panel.

Line numbers
If this option is on, TextWrangler displays line numbers along the left edge of the window.

Gutter
When this option is on, TextWrangler displays the gutter (see page 87). You can show or hide the gutter independently for each text window. This option is on by default.

Page Guide at N characters
When this option is on, TextWrangler displays the page guide at the specified character width. The page guide is a visible boundary indicator, whose color and contrast you can adjust (see page 191). This option is on by default.

Guide Contrast
You can use this sliding control to adjust the contrast level of the page guide display region. (See “Tab stops” on page 179.)

Text Status Bar
When any of the listed options are on, TextWrangler displays the status bar (see page 71). You can show or hide the status bar independently for each text window.

Cursor position
When this option is on, TextWrangler displays the current location (line and column) of the insertion point, or the endpoint of the current selection range in the status bar (see page 71).

Language
When this option is on, TextWrangler displays the Language popup menu in the status bar (see page 71).

Text encoding
When this option is on, TextWrangler displays the Text Encoding popup menu in the status bar (see page 71).

Line break type
When this option is on, TextWrangler displays the Line Break Type popup menu in the status bar (see page 71).

Document lock state
When this option is on, TextWrangler displays an padlock icon in the status bar to represent the document’s current lock state. You can click on the padlock to lock (or unlock) the current document (if possible).

Document save date
When this option is on, TextWrangler displays the current document’s last saved date and time (if applicable) in the status bar.

Document statistics

When this option is on, TextWrangler displays an item in the status bar which shows the number of characters, words, and lines in the document (and, if there's a selection, the number of characters, words, and lines in the selection range).

List Display Font

This option controls the font and size used to display text in browser list panes, including disk browser, search results browsers, etc. To change this option, click Set to bring up the standard Font panel, and choose the desired font and size. The default setting is 11 point Lucida Grande.

Application Preferences

The Application preferences control how TextWrangler checks for updates, when open files are verified, what action TextWrangler performs at startup, and various other global settings.

Open documents into the front window...

This option controls whether TextWrangler should attempt to open newly created or opened documents into the frontmost window (if possible), or whether each document should open directly into a separate text window.

This option is active by default, and while it is, TextWrangler will handle documents in the following manner.

When you open an existing document, TextWrangler will open the document into the frontmost editing window (and bring that window to the front if it is not already there.)

When you create a new document (via the File menu), TextWrangler will either use the frontmost editing window (if one is available), or make a new editing window if necessary.

Automatically refresh documents as they change on disk

This option controls whether TextWrangler checks if documents (files) have changed on disk while they’re open. If an open document has changed on disk, and there are no unsaved changes, TextWrangler will automatically reload the document. If a document has changed on disk and also has unsaved changes, TextWrangler will ask whether you want to reload the document from disk or keep the unsaved changes. This option is on by default.

The effects of the Revert command (from the File menu), and of a file Reload (which occurs when a document is reloaded by a refresh action) are both undoable.

Remember the N most recently used items

This text field lets you choose how many files appear on the Open Recent sub-menu of the File menu, and how many folders appear on the folder search popup menu in the Find Differences folder lists.

Always Show Full Paths in "Open Recent" Menu

Check this option to have TextWrangler always display full paths in the Open Recent menu. If this option is off, TextWrangler will only display path info when it's needed to distinguish between files with the same name.

When TextWrangler becomes active

This preference controls what TextWrangler does when you launch it, or activate it when there are no open windows (e.g. by clicking its Dock icon while the application is already running). To override any of these actions when launching TextWrangler, hold down the following modifiers.

Modifier(s)	Function
Option	Suppress startup items only
Shift	Disable all external services and startup items, and skip reopening all documents **except** those which contain unsaved changes.
Command-Control-Shift	Disable all external services and startup items, and optionally discard auto-recover information (which will result in the loss of any unsaved changes).

Do Nothing

Choose this option to prevent TextWrangler from opening a new text editing window.

New text document

Choose this option to have TextWrangler open a new, empty text editing window.

Reopen documents that were open at last quit

When this option is on, TextWrangler will remember what documents (as well as disk browsers and FTP/SFTP browsers) were open when you choose the "Quit" command, and will attempt to reopen those documents the next time you launch it. This option is on by default.

Restore unsaved changes

When this option is on, TextWrangler will preserve the contents of any unsaved document contents when you quit (including untitled documents) and restore those documents the next time you launch it. If you prefer the traditional Quit behavior, turn this option off.

Include documents on servers

When this option is on, TextWrangler will attempt to reopen documents from remote servers when you launch it.

Automatically check for updates

This option controls whether TextWrangler automatically looks to see if a newer version is available. Regardless of the setting of the checkbox, you can manually check for an update at any time by clicking the Check Now button.

The version checking mechanism used by TextWrangler protects your privacy. It works by requesting information about the currently available version from Bare Bones Software's web server. The server will log the date, time and originating address of the request, and which versions of the OS and TextWrangler you are using. This information is used to guide the future development of TextWrangler; it is not personalized and will not be disclosed. Click the Privacy button to view our posted privacy policy.

This option is not present in copies of TextWrangler obtained via the Mac App Store.

Editing Preferences

The Editing preferences control the behavior of various general editing behaviors.

Highlight instances of selected text

When this option is on and you make a selection (that doesn't consist entirely of whitespace or punctuation), TextWrangler will underline all occurrences of the selection within the current document, and you can use the Search menu commands Next Occurrence of <string> and Previous Occurrence of <string> to navigate the occurrences. This option is on by default.

You can also control the duration of the delay before TextWrangler applies highlighting via the "Delay" control below this option.

Note Any navigation you undertake via this feature is independent from the Find and Live Search commands, except that TextWrangler will add the selected string to its search history for convenient future reuse.

Use "hard" lines in soft-wrapped views

When this option is on, the line number bar, cursor position display, and Line Number commands in editing views will use line and character position numbers that correspond to the "hard" line breaks actually present in the document, rather than the soft-wrapped line breaks.

Additionally, when this option is on, line selection commands and gestures, including the Select Line command, triple-clicking, and click selection in the left margin, will treat only "hard" line breaks as line boundaries.

Soft-wrapped line indentation

This option lets you specify how TextWrangler should indent soft wrapped text: flush with the left edge of the window, at the same indent level as the first line of the paragraph, or indented one level deeper than the first line of the paragraph.

Line spacing

This control allows you to adjust the amount of space between lines of text in editing views. The default value is consistent with previous versions of TextWrangler.

Editor Defaults Preferences

The Editor Defaults preferences control the behavior of newly created document windows and documents without saved state information. Many of the options in this panel parallel options provided in the Text Options sheet and in the Text Options popup in the toolbar. The difference is that the options in the Text Options sheet and the Text Options popup control only the behavior of the active window, while the Editor Defaults preferences control the behavior of all new windows.

Auto-indent

When this option is selected, pressing the Return key in new windows automatically inserts spaces or tabs to indent the new line to the same level as the previous line.

Tip To temporarily invert the sense of the Auto Indent option while typing, hold down the Option key as you press the Return key.

Balance while typing

When this option is selected, TextWrangler flashes the matching open parenthesis, brace, bracket, or curly quote when you type a closing one. This option is useful when you are editing source files, to ensure that all delimiters are balanced.

Use typographer's quotes

When this option is on, TextWrangler will automatically substitutes curly (or typographer's) quotes (“ ” ‘ ’) for straight quotes (" ') in any new documents you create.

Tip To type a straight quote when this option is selected (or to type a curly quote when the option is deselected), hold down the Control key as you type a single or double quote.

Note You should avoid using typographer's quotes when creating or editing any plain-text documents such as email message content or source code.

Auto-expand tabs

When this option is on, TextWrangler inserts an appropriate number of spaces instead of a tab character every time you press the Tab key.

Show invisible characters

This option shows or hides non-printing characters in the window. Select this option when you want to see line breaks, tabs, and gremlins (invisible characters). TextWrangler uses these symbols to represent non-printing characters:

Symbol	Meaning
∆	tab
◊	space
•	non-breaking space

Symbol	Meaning
¬	line break
¶	page break
¿	other non-printing characters

Show Spaces

When this option is on (and Show Invisibles is also active), TextWrangler will display placeholder characters for spaces. Turn this option off to suppress the display of spaces (reducing visual clutter when you are displaying invisible characters).

Note Non-breaking spaces (typed by pressing Option-space) will not be displayed with a placeholder.

Check spelling as you type

When this option is on, TextWrangler will automatically check spelling as you type, and underline any potentially misspelled words. Turn this option off to prevent TextWrangler from automatically checking spelling.

You can turn on automatic spell checking for the active document only by choosing Check Spelling as You Type from the Text menu. (See “Check Spelling As You Type” on page 100.)

Default font

This option controls the standard font and font size which TextWrangler uses to display the contents of text windows. To change this option, click Set to bring up the standard Font panel, and choose the desired font and size. The default font is “Menlo Regular” at 12 points.

Note You can also adjust the default tab width on a per-language basis. To do so, select a language entry in the Languages preference panel, click “Options” to bring up the language options sheet, and enter the desired tab width in the Editing section of this sheet.

Tab Width

This option controls the default number of spaces that TextWrangler uses to represent the width of a tab character.

Soft Wrap Text

When this option is selected, TextWrangler soft-wraps the text in the file to the right margin that you choose: the Page Guide, the window width, or a specified number of characters, as selected by the options below the checkbox.

Keyboard Preferences

The Keyboard preferences control TextWrangler's response to the use of various special keys, including the ability to recognize Emacs key bindings

"Home" and "End" Keys

There are three potential choices for this option:

Scroll to Beginning and End of Document

Choose this setting to have the Home and End keys perform these respective actions. This is the default setting, which reflects the standard key motion behavior in Macintosh applications.

Move Cursor to Beginning and End of Current Line

Choose this setting to have the Home and End keys perform these respective actions instead. This option may be useful for those accustomed to Windows editing key behavior.

Progressive (BRIEF Compatible)

Choose this option to have the Home and End keys behave as follows on successive presses:

- the first press will move the insertion point to the beginning (or end) of the current line;
- the second press will move the insertion point to the beginning of the first line (or the end of the last line) in the current page of text, without scrolling;
- the third press will move the insertion point to the beginning (or end) of the document.

The behavior is progressive within a specific time period. After the period expires, or if you change the selection range by other means, the behavior state resets, so the next press of Home or End will behave as in the first step described above.

The factory default timeout period is ten seconds. (There is an expert preference option to control this period; please see the "Expert Preferences" page of TextWrangler's online Help for details.)

Enter key generates Return

When this option is on, TextWrangler will generate a carriage return when you press the Enter key.

When this option is off, pressing the Enter key will bring the current insertion point (or selection range) into view.

Allow Tab key to indent text blocks

When this option is on, you can press the Tab key to invoke the Shift Right command, or Shift-Tab to invoke the Shift Left command; this may be useful for those accustomed to Windows editing key behavior. When this option is off, pressing Tab will insert a tab character in the normal manner. This option is off by default.

Enable Shift-Delete for forward delete

When this option is on, holding down the Shift key with the Delete key makes the Delete key work the same way as the Forward Delete key on extended keyboards.

When auto-indenting, remove leading white space from indented line

When this option is on, if the current line is not indented but there is whitespace followed by text to the right of the insertion point, when you type Return, TextWrangler will insert a line break and strip the following whitespace, leaving the remaining text also left-aligned.

Allow Page Up and Page Down keys to move the insertion point

When this option is off, the Page Up and Page down keys function in their usual fashion. Refugees from other platforms may feel more comfortable if this option is turned on; doing so allows the insertion point to remain in the same position relative to the top and bottom of the window (which entails changing its position in the text) when scrolling via Page Up or Page Down.

Option-¥ on Japanese keyboards

This option controls whether typing Option-yen on a Japanese keyboard generates a yen symbol " ¥ " or a backslash "\".

Emulate Emacs key bindings

If turned on, this option allows you to use the basic Emacs navigation keystrokes to move around in editing views. It is not a full Emacs emulation mode; rather, it is more of a comfort blanket for individuals with Emacs key bindings hard-wired into their muscle memory. See Appendix B, "Editing Shortcuts," for a list of the Emacs commands TextWrangler supports.

Display status window

When both this option and "Emulate Emacs key bindings" are on, TextWrangler will display a small palette which shows Emacs shortcuts as you type them.

Languages Preferences

The Languages preferences allow you to configure how TextWrangler maps file names to language types (e.g. ".html" to HTML), and allows you to apply customized behavior and display parameters to any installed language.

Installed Languages

Click the "Installed Languages" button at the bottom of this panel to see a complete list of installed languages, together with the language module version number (if applicable) and filename extension(s) associated with each language. (This list includes both languages intrinsically supported by TextWrangler, and those added via installed language modules.)

By default, TextWrangler will apply your active preference settings within each language.

If you wish to modify how TextWrangler treats documents having a particular language, e.g. to have TextWrangler use a specific tab width or a custom color scheme, you may add a custom language preference.

To create such a preference, click the plus (+) button below the list of Custom Language Preferences, and select the desired language from the resulting popup. When you do so, TextWrangler will display a language options sheet which contains the following sections:

- General: In this section, you can view or change the comment-start and comment-end strings used by the Un/Comment command on the Text menu for the selected language, or to view or change the Reference URL Template used by the Find in Reference command.
- Editor: In this section, you can view or change the default display and editing options used for documents in the selected language. (These options parallel the options provided by the Text Options command.)
- Display: In this section, you can view or change the default items which appear on the navigation bar and status bar for documents in the selected language. You can also choose any available color scheme to use for syntax coloring of documents in the selected language.

To remove an existing language preference, select the desired entry in the list of Custom Language Preferences, and click the minus (-) button below the list. Once you have removed the entry, TextWrangler will again apply its active global preferences settings to all documents with that language.

Custom Extension Mappings

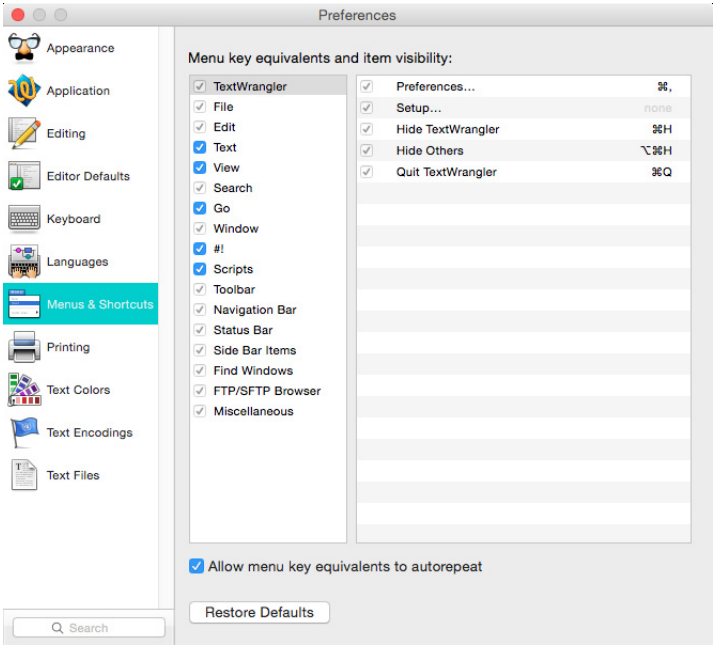
TextWrangler includes a set of default file extension mappings which cover the most common usages for each supported language, while each language module ordinarily contains extension mappings for the language it supports.

You may add (or remove) additional extension mapping via the Custom Extension Mappings list. To add a mapping, click the plus (+) button below the list, click in the Suffix column and type the desired filename extension, then select the associated language via the adjacent popup. (You can also edit existing mappings in the same manner.)

Note You can use wildcards in the suffix to indicate single characters (?), any number of characters (*), or a single digit (#). For example, "page.#html" could map to a different language from ".html".

Menus & Shortcuts Preferences

The Menus & Shortcuts preferences allow you to show or hide whole menus or individual commands. You can also assign key equivalents to commands and various window elements, as well as to clippings and scripts.



Menu Key Equivalents and Item Visibility

This section of the preference panel displays a hierarchical list of each menu and menu command available within TextWrangler.

You can hide any menu or command which is not necessary for TextWrangler to function, by turning off the checkbox next to that item's name. (The checkbox is disabled for necessary items, such as the File menu and the Quit command.)

You can assign or change the keyboard shortcut (key equivalent) for any menu command, as well as items on the Text Options, Markers, and Line Breaks toolbar popup menus, by double-clicking on the right-hand portion of that command's list item and typing the desired key equivalent.

To clear the key equivalent from a menu command, double-click on the right-hand portion of that command's list item and press the Delete key.

Click Restore Defaults to restore all key equivalents to their factory default values (as listed in Appendix A).

Available Key Combinations

All menu key combinations must include either the Command key or the Control key (or both), except function keys, which may be used unmodified. The Help, Home, End, Page Up and Page Down keys can be used in menu key combinations as well. The Help key can be assigned without modifiers; the others must be used in combination with at least either the Command or Control key.

Note The system may preempt certain key combinations, such as Command-Tab.

Allow menu key equivalents to autorepeat

Turn this option on to enable autorepeat when typing key equivalents. (This option is off by default since according to the Macintosh Human Interface Guidelines, menu commands should not autorepeat.)

Printing Preferences

The Printing preferences control TextWrangler's default document printing behavior.

Print using document's font

When this option is on, TextWrangler uses the document's display font and tab settings when printing.

Printing font

This option specifies the default font TextWrangler uses for printing when "Print using document's font" is not active. Click Set to bring up the standard Font panel, where you can choose a font and font size. The current printing font options appear in the display box.

Frame printing area

When this option is on, TextWrangler draws a box along the edges of the printed text.

Print page headers

When this option is on, TextWrangler prints the page number, the name of the file, the time and date printed in a header at the top of each page.

Print full pathname

When this option is on, TextWrangler prints the full pathname of the file being printed in the header.

Time stamp

This option let you choose whether the date that appears in the printed page header is the date that the file was last modified or the date that the file was printed.

Print line numbers

When this option is on, TextWrangler prints line numbers along the left edge of the paper.

1-inch Gutter

When this option is on, TextWrangler leaves a one-inch margin along the left edge of the paper. Use this option if you usually store printed pages in three-ring binders.

Print color syntax

If this checkbox is on, TextWrangler prints all colorized text within the document in color. You should generally use this option only on color printers, as colorized text may come out in difficult-to-read dithered shades of gray on black-and-white printers.

Text Colors Preferences

The Text Colors preferences let you adjust the default colors that TextWrangler applies to both general and language-specific syntax elements, as well as various standard elements including the foreground and background text colors, spelling errors, spaces and invisible characters, and highlight colors.

Selecting and Saving Color Schemes

TextWrangler now offers several built-in color schemes, which you may use as-is or as the basis for a custom scheme. To save a customized color scheme, click the "Save Scheme..." button and name the theme. To load a saved color scheme, choose it in the Color Scheme popup menu. (As in prior versions, any color scheme you create will be stored in the "Color Schemes" subfolder of TextWrangler's application support folder.)

TextWrangler will also import any BBColors files which you place in the "Color Schemes" folder, and convert them.

You can further associate a saved color scheme with any language via the Custom Language Preferences list in the Languages preference panel. (See "Languages Preferences" on page 186.)

How to Change an Element's Color

The color bars show the colors that TextWrangler uses to display different interface and language elements. To change the color for any element, click the adjacent color box to open the system color picker which you can use to select a new color. To restore all colors and options to their default settings, click the Restore Defaults button.

Language-Specific Colors

The center section of the Text Colors panel contains groups of language-specific syntax coloring options, which you can adjust to specify the colors TextWrangler uses to display the corresponding language elements.

Though the available set of languages and elements is too extensive to list in total, here are some common elements:

- **Comments** include all text set off by a language's designated comment marker(s).
- **Strings** (and **Numbers**) are defined by each individual language's specification.
- **Language keywords** are those terms defined in a language's specification
- **Predefined symbols** are terms which are not language keywords, but which are predefined by a language's reference implementation, or which are part of a language's standard library/framework support, or which have other special meaning to developers writing code in that language.
- **ctags symbols** are any words or elements identified in an associated ctags file.

while in general, most elements' natures should be clear from their display names (e.g. **Preprocessor directives**).

Global Colors

The bottom section of the Text Colors panel contains options which control global colors used within any language such as the foreground (text) and background (window) colors and the color of the underline used by the spelling checker to mark questioned words.

Background

This option controls the background color used within editing windows (and other content display views).

Misspelled words

This option controls the color TextWrangler uses to highlight misspelled words.

Spaces

This option controls the color TextWrangler uses to display spaces when the Show Invisibles and Show Spaces display options are active.

Other invisibles

This option controls the color TextWrangler uses to display invisible characters other than spaces when the Show Invisibles display options is active.

Use custom highlight colors

Turn this option on to have TextWrangler use custom highlight colors. You can choose the primary and secondary highlight colors.

Highlight insertion point

When this option is on, TextWrangler highlights the line currently containing the insertion point using the indicated color. You can choose the line highlight color.

Text Encodings Preferences

The top of the Text Encodings preference panel contains an alphabetical list of every character set encoding available in the system, and allows you to choose which of these encodings TextWrangler includes in its menus. These menu are:

- The Read As popup menu in the Open dialog

- The Encoding popup menu in the Options dialog within the Save dialog
- The Encoding popup in the status bar
- The character set popup menus in various HTML tools dialogs (including the New HTML Document dialog)
- The encoding selection popup menus in this preference panel

To include an encoding for display, select it and click Enable. To remove an encoding from display, select it and click Disable. To include all encodings or remove all but the required the encodings, click the Enable All or Disable All buttons respectively.

(All available Unicode encodings are permanently enabled and cannot be turned off.)

Tip To keep the length of the encoding menus manageable, you should add only those encodings which you use frequently.)

Default text encoding for new documents

TextWrangler uses the encoding specified by this option for new documents which do not contain an intrinsic encoding specification.

If file's encoding can't be guessed, try

If TextWrangler cannot determine a file's proper encoding by examination, it will try opening the file using the encoding(s) contained in this list, in the order they appear.

Text Files Preferences

The Text Files preferences control how TextWrangler opens and saves files, including whether to make backups.

Line breaks

This option controls what kind of line breaks TextWrangler writes when creating a new file. You can choose:

- Unix line breaks (ASCII 10) for general use. This is the default option.
- Classic Mac line breaks (ASCII 13) if you will be using the file with Classic Macintosh applications.
- Windows line breaks (ASCII 13/10) if the file will reside on a Windows server or if you are sending it to someone who uses a Windows system

Ensure file ends with line break

When this option is on, TextWrangler will add a line break at the end of the file if there is not already one present.

You can also adjust this option on a per-language basis by adding custom language preferences. (See "Languages Preferences" on page 186).

Strip trailing whitespace

When this option is on, TextWrangler will trim all trailing non-vertical whitespace from the document file before writing it out.

You can also adjust this option on a per-language basis by adding custom language preferences. (See “Languages Preferences” on page 186).

Backups

These options control whether TextWrangler should make backup copies of edited files, and the manner in which it does so.

Make backup before saving

Turn this option on to have TextWrangler automatically make a backup copy of each file that you save. TextWrangler creates a single backup file for each file that you save in the same folder as that file. This option is global and backups can no longer be made on a per-file basis. However, you can exclude individual files from being backed up by adding an Emacs variable to them (see “EditorConfig” on page 38).

When this option is on, and you close a document with unsaved changes and elect to discard those changes (“Don't Save”), TextWrangler will automatically save a snapshot of the document's contents in the same directory as the document, and the snapshot file’s name will follow the Emacs convention “#foo.txt#” (or if the “Preserve file name extension” (see below) is on, the snapshot's name will be “#foo#.txt”).

Keep historical backups

When this option is on, TextWrangler will preserve backups in the folder “~/Documents/TextWrangler Backups/” and the “Preserve File Name” option (see below) will automatically be turned on and locked.

Within the backup folder will be one folder for each day's backup files. The format of the dated folder name is static and non-localized: YYYY-MM-DD. Inside of each day’s backup folder will be all of the backup files made on that day, each named using a timestamped format.

You may change the location of the backup folder by placing a folder alias named “TextWrangler Backups” in your “Documents” folder (~/Documents/) and TextWrangler will follow the alias.

Preserve file name extension

By default, the backup files which TextWrangler creates are named in accordance with current system conventions (which themselves follow the old Emacs convention): the backup file takes the name of the original with a tilde appended; for example, “foo.html~” is the backup of “foo.html”.

If you want backup files to have the same filename extension as the originals, turn on this option to have TextWrangler place the tilde after the “base” name of the file; for example, “foo~.html”.

Controlling Backups with Emacs Variables

You may also use an Emacs variable to control whether or not a given file is backed up. There are two ways to do this:

Absolute: If the variable line/block contains a “make-backup-files” variable, that variable’s value will override the global “Make Backup Before Saving” preference.

```
-*- make-backup-files: 1 -*-    --> always back up this file
-*- make-backup-files: 0 -*-    --> never back up this file
```

If the first letter of the variable’s value is “y”, “t”, or “1”, the value is “yes”, otherwise it’s “no”. These are all synonymous:

```
make-backup-files: yes
make-backup-files: y
make-backup-files: true
make-backup-files: t
make-backup-files: 1
```

Inhibit: If the variable’s line/block contains a “backup-inhibited” variable, and its value is true (see above), then the file will never be backed up, even if “Make backup before saving” is turned on in the global preferences.

Expert preferences settings

In addition to the preference settings which can be made through the Preferences window, TextWrangler supports a number of expert preferences which you can adjust by issuing an appropriate “defaults write” command.

The “Expert Preferences” page within TextWrangler’s built-in Help book (choose “TextWrangler Help” from the “Help” menu) contains a complete, current listing of these options.

The Setup Window

The Setup window allows you to manage several types of configuration info which TextWrangler uses, including FTP/SFTP bookmarks, file filters, and grep search patterns. (In versions prior to 4.0, most of this information was managed through the Preferences window.)

Bookmarks

This panel lists any bookmarks you have created for FTP and SFTP servers. You may click the plus (+) button to create a new bookmark, double-click any bookmark item to edit its stored options (or rename it), or select a bookmark and click the minus (-) button to remove it.

Filters

The Filters panel lists all the file filters you have defined for use with multi-file searches, Find Differences, and disk browsers. You may click the plus (+) button to create a new filter, double-click any filter item to edit its stored options (or rename it), or select a filter and click the minus (-) button to remove it. (For more information on using file filters in searches, see Chapter 7.)

Patterns

This list displays all the grep patterns (regular expressions) you have stored via the Grep pattern popup in the Find and Multi-File Search windows. These patterns are also available in most commands which allow you to specify grep patterns, such as the Process Lines commands in the Text menu.

You may click the plus (+) button to create a new pattern, double-click any pattern item to edit its stored options (or rename it), or select a pattern and click the minus (-) button to remove it.

CHAPTER

11 Scripting TextWrangler

TextWrangler offers access to nearly all of its features and commands via AppleScript. This chapter provides a brief overview of AppleScript, discusses TextWrangler's scripting model, and explains how you can use scripts within TextWrangler.

An excellent way to learn how to script TextWrangler is to look at the scripts others have written for it, or to turn on recording in your script editor while you perform actions in TextWrangler. A number of example scripts are included in the standard distribution package. The TextWrangler Talk discussion group is also a good resource for learning more about scripting.

```
http://groups.google.com/group/textwrangler/
```

IMPORTANT Regardless of whether you are new to scripting TextWrangler or are familiar with scripting previous versions, we strongly recommend that you carefully review the sections "TextWrangler and AppleScript" and "Working with Scripts" in this chapter.

In this chapter

AppleScript Overview **197**
About AppleScript – 198
Scriptable Applications and Apple Events – 198
Reading an AppleScript Dictionary – 199
Recordable Applications – 204 • *Saving Scripts* – 205
Using Scripts with Applications – 205 • *Scripting Resources* – 206
Using AppleScripts in TextWrangler **207**
Recording Actions within TextWrangler – 207 • *The Scripts Menu* – 208
The Scripts Palette – 209 • *Organizing Scripts* – 209
Attaching Scripts to Menu Items – 210
Attaching Scripts to Events – 211
TextWrangler's Scripting Model **216**
Script Compatibility – 216 • *Getting and Setting Properties* – 218
Performing Actions – 219 • *Common AppleScript Pitfalls* – 224

AppleScript Overview

If you are familiar with AppleScript, you should have little difficulty scripting TextWrangler. It has a robust and highly flexible object model. If you do not know much about scripting, though, read on for an introduction to the necessary concepts.

About AppleScript

AppleScript is an English-like language which you can use to write scripts that automate the actions of applications, and exchange data between applications. Although AppleScripts can manipulate applications' user interfaces by taking advantage of the system's GUI Scripting capability, this is not their primary function. Rather, scripts talk directly to a application's internals, bypassing its user interface and interacting directly with its data and capabilities.

If you want to insert some text into a document, emulating a user typing into an editing window is not the most efficient way of accomplishing this. With AppleScript, you just tell the application to insert the text directly. If you want the application to save the frontmost document, you need not mime choosing Save from the File menu, but rather just tell the application to save its frontmost document.

Note AppleScript is actually a specific language which resides atop the general Open Scripting Architecture (OSA) provided by Mac OS X. Although AppleScript is by far the most common OSA language, there are others, including a JavaScript variant. All OSA languages are capable of accomplishing similar things, although the actual commands used differ from one language to the next. In this chapter, we will focus exclusively on AppleScript, since it is the standard scripting language, but you should bear in mind that there are other options.

Scriptable Applications and Apple Events

Since AppleScripts must have direct access to an application's internal data structures, any application that will be used in an AppleScript must be designed to allow this access. We say such applications are *scriptable*. TextWrangler is scriptable, as are many, many other programs. However, it is important to note that not *every* application is scriptable, and AppleScripts are not the best solution for automating applications that are not.

What goes on in an application that is scriptable? The foundation of AppleScript is something called the *Apple Event*. Macintosh applications are designed around an event loop; they go around in circles waiting for you, the esteemed user, to do something (choose a menu command, press some keys, and so on). These actions are passed to the application by the operating system in the form of an *event*. The application decodes the event to figure out what you did, and then performs an appropriate operation. After an event has been handled, the application goes back to waiting for another one. (At this point, the Mac OS may decide to give some time to another application on your computer.)

Apple Events are special events that applications send to each other, enabling a feature called *inter-application communication* (IAC). (It's a mouthful, but it just means applications can talk to each other.) Apple Events are also the way AppleScripts tell applications what to do, and which data to retrieve. So to be scriptable, an application must first support Apple Events.

Apple Events in their naked form are raw and cryptic things—bits of hieroglyphics only a programmer could love. So a scriptable application also has a *scripting dictionary*. The scripting dictionary tells any application that lets you write AppleScripts, such as the standard Script Editor, the English-like equivalent for each Apple Event and each event's parameters.

It is important to note that because Apple Events were originally designed to allow applications to communicate with each other, AppleScripts automatically inherit the ability to talk to more than one application. It is common in the publishing industry, for instance, to write scripts that obtain product information from a FileMaker Pro database and insert it into an InDesign file. This integration is one of the Macintosh's primary strengths.

You use AppleScript's *tell* verb to indicate which application you are talking to. If you are only sending one command, you can write it on one line, like this:

```
tell application "TextWrangler" to count text documents
```

If you are sending several commands to the same application, it is more convenient to write it this way:

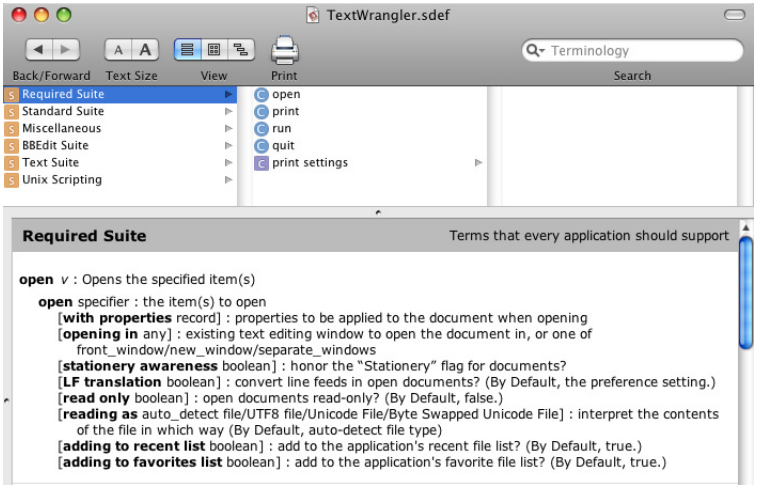
```
tell application "TextWrangler"
    count text documents
    repeat with x from 1 to the result
        save text document x
    end repeat
end tell
```

The Script Editor automatically indents the lines inside the *tell* block for you so you can more easily follow the organization of the script.

Reading an AppleScript Dictionary

To display an application's AppleScript dictionary, you can simply drag that application onto the Script Editor icon, or use the Script Editor's Open Dictionary command. As we noted earlier, all scriptable applications include a dictionary that tells AppleScript how to convert English-like commands into the Apple Events actually expected by the application. The Script Editor uses this same information to display a sort of "vocabulary guide" that helps you write your scripts.

We will naturally use TextWrangler's dictionary, shown below, to illustrate how to read a dictionary.


TextWrangler.sdef
Back/Forward
Text Size
View
Print
Terminology
Search
Required Suite
Standard Suite
Miscellaneous
BBEdit Suite
Text Suite
Unix Scripting
open
print
run
quit
print settings
Required Suite
Terms that every application should support
open v : Opens the specified item(s)
open specifier : the item(s) to open
[with properties record] : properties to be applied to the document when opening
[opening in any] : existing text editing window to open the document in, or one of
front_window/new_window/separate_windows
[stationery awareness boolean] : honor the "Stationery" flag for documents?
[LF translation boolean] : convert line feeds in open documents? (By Default, the preference setting.)
[read only boolean] : open documents read-only? (By Default, false.)
[reading as auto_detect file/UTF8 file/Unicode File/Byte Swapped Unicode File] : interpret the contents
of the file in which way (By Default, auto-detect file type)
[adding to recent list boolean] : add to the application's recent file list? (By Default, true.)
[adding to favorites list boolean] : add to the application's favorite file list? (By Default, true.)

(You will probably want to make the window bigger if you have room on your screen.)

Down the left side is a list of every event and object supported by the application. An event is a verb—it tells the application what to do. A class is a noun: a piece of data, or a structured collection of data, inside the program. In TextWrangler, for instance, classes are things like files, windows, the clipboard, browsers, and so on.

Suites

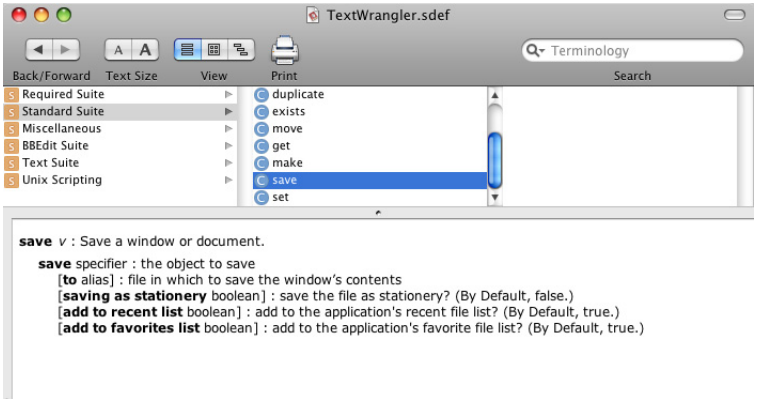
The first thing you will notice is that the events and classes are divided into *suites*. A suite is just a collection of related events and classes. Apple, for instance, has decreed that all applications should support particular events, which together are called the Required Suite. Another Apple-defined suite is the Standard Suite: if an application offers certain functions which Apple considers to be common, it should use these standard terms, so that scripters do not need to learn a new term for each application they work with. After that, it is a free-for-all—each developer is free to organize their events and classes however they think best.

In addition to the Required and Standard suites, TextWrangler has a Miscellaneous suite, a TextWrangler Suite, a Text suite, and a Unix Scripting suite.

Within each suite, events—verbs—are displayed in normal text, while classes—nouns—are italicized. Most commands sent to TextWrangler will start with one of the verbs. (In some cases, *get* might be implied.)

Events

Let's look more closely at one of the events—*Save* is a good one to start with. It is shown below.



The right side of the window shows the syntax of the selected event, as well as a brief description of its function. The boldface words are keywords; they must be included exactly as shown or the script will not compile. The normal text tells you what kind of information goes after each keyword. For example, after *save* you must give a reference; the italicized comment next to that line indicates that it is a reference to the window to be saved. In other words, some window object, which in TextWrangler would be *window 1* for the frontmost window, or *window "Text File"* if you want to specify a window by name. (we will show you how to figure all that out in a moment—you have to look at the window class's dictionary entry.)

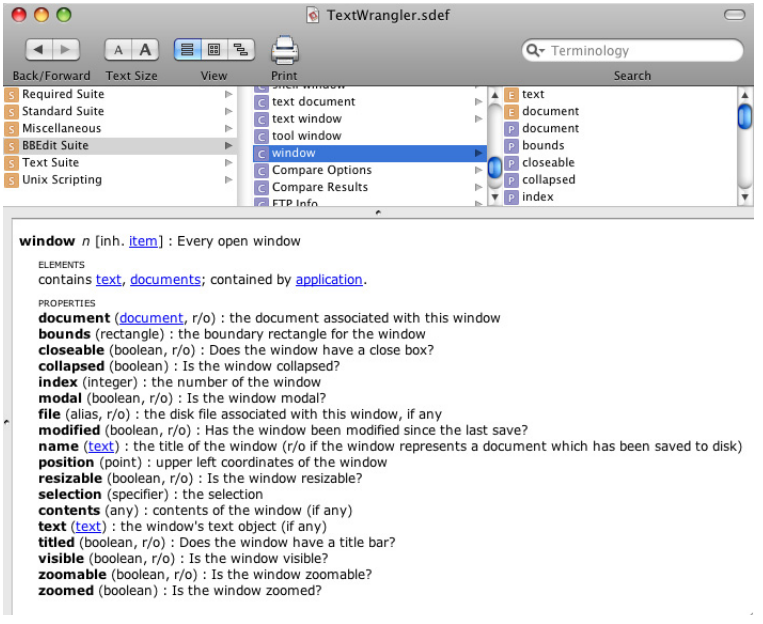
Anything in square brackets is optional. Most of the rest of the *save* event is optional, in fact. The basic event just saves the frontmost window to the same file from which it was opened. However, you can also optionally include the word *to* followed by a file reference. (You specify a file simply by using the word *file* followed by the path name of the file, as in *file "Hard Disk:Users:Adam:Documents:My file".)* If you specify a file to save the window to, the text will be saved into that file instead of the file it came from—like using Save As instead of Save.

The last three optional parts of the *save* event are denoted as boolean. That means they take either a true or a false value. In AppleScript, there are a couple of different ways to specify boolean values. You can write *saving as stationery true* to tell TextWrangler to save the file as a stationery document. Or you can write *with saving as stationery.* You will notice that the last two parameters default to true if you do not specify them as false. To do that, you would use *add to recent list false* or *without add to recent list*. Whichever way you write it, you will notice that when you compile the script, AppleScript rewrites it using "with" or "without". Since that is the syntax AppleScript seems to like best, that is probably the one you should get used to thinking in.

Let's take a look at another one: the prosaic *get.* Select *get* from TextWrangler's dictionary listing and take a quick look at its class definition.You use *get* to retrieve information from an application. You must specify a reference to the object you want to retrieve, and you can specify a *coercion*—a condition that tells AppleScript to treat one type of data as if it were another—by adding the *as* clause. However, after that is the *Result:* line, which we have not seen before. This line tells you what type of value the command returns. (This value is placed in the AppleScript system variable called *the result.) Get* can retrieve any kind of object, so it can return anything, as indicated here. Other events might return a specific type of result, or none at all. *(Save* did not have a *Result:* line in its dictionary entry, which means it does not return a result.)

Classes and the Class Hierarchy

Let's look now at a typical class definition: *window* will do nicely. It is in the TextWrangler Suite, toward the bottom.


TextWrangler.sdef
Back/Forward
Text Size
View
Print
Terminology
Search
Required Suite
Standard Suite
Miscellaneous
BBEdit Suite
Text Suite
Unix Scripting
text document
text window
tool window
window
Compare Options
Compare Results
text
document
document
bounds
closeable
collapsed
index
window n [inh. item] : Every open window
ELEMENTS
contains text, documents; contained by application.
PROPERTIES
document (document, r/o) : the document associated with this window
bounds (rectangle) : the boundary rectangle for the window
closeable (boolean, r/o) : Does the window have a close box?
collapsed (boolean) : Is the window collapsed?
index (integer) : the number of the window
modal (boolean, r/o) : Is the window modal?
file (alias, r/o) : the disk file associated with this window, if any
modified (boolean, r/o) : Has the window been modified since the last save?
name (text) : the title of the window (r/o if the window represents a document which has been saved to disk)
position (point) : upper left coordinates of the window
resizable (boolean, r/o) : Is the window resizable?
selection (specifier) : the selection
contents (any) : contents of the window (if any)
text (text) : the window's text object (if any)
titled (boolean, r/o) : Does the window have a title bar?
visible (boolean, r/o) : Is the window visible?
zoomable (boolean, r/o) : Is the window zoomable?
zoomed (boolean) : Is the window zoomed?

All windows in TextWrangler belong to this class. A class defines a particular kind of object; a particular example of an object belonging to the class is said to be an instance of that class, or just an object of that class. So here we are looking at the class itself; each individual window object has all these properties.

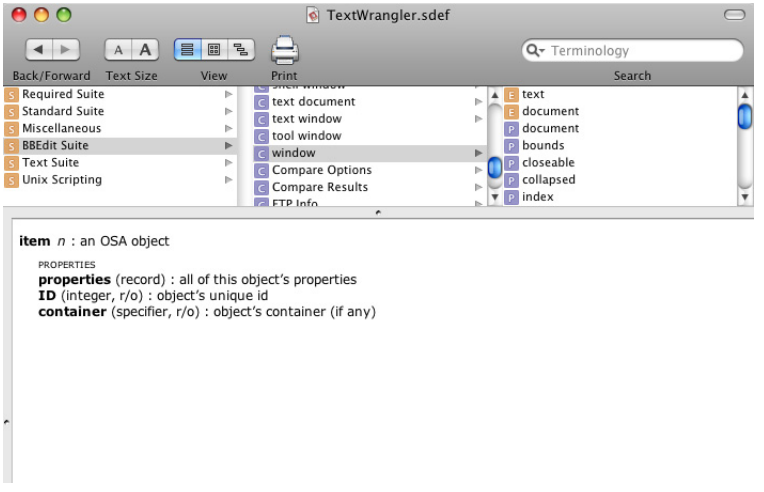
After a tag line that tells you about the class ("an open window") comes the plural form. AppleScript lets you refer to windows either singly or as a group, so it needs to know what the plural of every term is. For example, try this little script:

```
tell application "TextWrangler" to count windows
```

The result of this script is the total number of window objects currently displayed by TextWrangler.

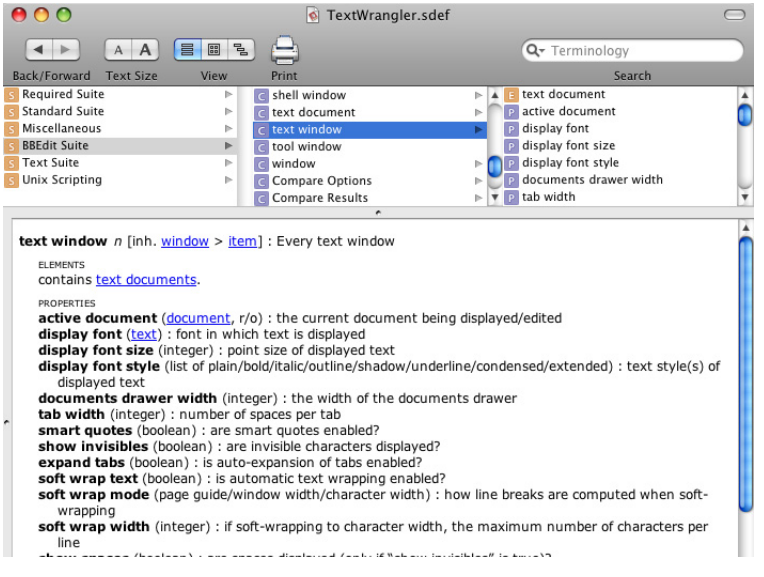
After the plural form comes a list of properties. Some objects do not have properties—for example, a string—but many applications do. An object's properties are merely a collection of data that describes that particular object. For example, as you look down the list of window properties, you will see that every window has a name, every window has a position, every window has bounds (the area of the screen it covers), and so on.

The first item on the list, though, is <*inheritance*> *item*. This tells you that a window is a kind of item, and that it therefore has all the properties of an item. Take a quick look at *item's* class definition, shown below.



You will see three properties: *properties, ID*, and *container*. The first entry *properties* is a record containing all the object's properties. In other words, because a window is an item, it has, in addition to all its listed properties, another property which returns all the other properties as a record—a single piece of data that can be stored in a variable. Every class in TextWrangler is part of a hierarchy with the *item* class at the top, so every object in TextWrangler "inherits" the *properties* property. This catch-all property can be handy for making exact duplicates of objects, among other uses.

You may realize that TextWrangler has several kinds of windows; you can see their classes listed in the dictionary: clipboard window, differences window, disk browser window, text window, tool window, and the like. Let's look at *text window:*


TextWrangler.sdef
Terminology
Back/Forward
Text Size
View
Print
Search
Required Suite
Standard Suite
Miscellaneous
BBEdit Suite
Text Suite
Unix Scripting
shell window
text document
text window
tool window
window
Compare Options
Compare Results
text document
active document
display font
display font size
display font style
documents drawer width
tab width
text window n [inh. window > item] : Every text window
ELEMENTS
contains text documents.
PROPERTIES
active document (document, r/o) : the current document being displayed/edited
display font (text) : font in which text is displayed
display font size (integer) : point size of displayed text
display font style (list of plain/bold/italic/outline/shadow/underline/condensed/extended) : text style(s) of displayed text
documents drawer width (integer) : the width of the documents drawer
tab width (integer) : number of spaces per tab
smart quotes (boolean) : are smart quotes enabled?
show invisibles (boolean) : are invisible characters displayed?
expand tabs (boolean) : is auto-expansion of tabs enabled?
soft wrap text (boolean) : is automatic text wrapping enabled?
soft wrap mode (page guide/window width/character width) : how line breaks are computed when soft-wrapping
soft wrap width (integer) : if soft-wrapping to character width, the maximum number of characters per line

You can see that a text window inherits all the properties of the *window* class. And, since the *window* class inherits all the properties of the *item* class, this means that the *text window* class also has the *properties* property defined by the *item* class.

To make explicit what you might have already gathered, classes in AppleScript form a hierarchy. That is, classes can be based on other classes. Such a class is called a *subclass,* and the class on which a subclass is based is referred to as its *parent class.* (In AppleScript, classes can only have one parent. Multiple inheritance is a feature found in more complex languages.)

The idea of a class hierarchy makes it easier for us to add new features to TextWrangler, since when we want to create a new kind of window, half the work is already done. However, when scripting, you may need to flip back and forth between two or more class definitions to find all the properties of the object you are working with. (This is, technically speaking, a limitation of Apple's Script Editor. There is no reason the inherited properties could not automatically be included in a subclass listing by a smarter editor, for example, Script Debugger, which does this.)

Now that we have the class hierarchy under control, let's look at the properties themselves more closely. we will stick with the *text window* class at this point.

Properties of an object are referred to using the preposition *of.* For example, the following line of script returns the font of the frontmost text window.

```
tell application "TextWrangler" to get display font of
text window 1
```

Note In this specific example, you can just write *get display font of window 1*. AppleScript will figure out that window 1 is more specifically a text window, and therefore has a *display font* property, even though the generic *window* class does not have any such property. All the properties of the object are available even if you did not use its specific class name. However, in most cases, you should specify exactly the object you want; this distinction is especially important when dealing with text documents (content) versus text windows (display elements).

You can set the properties using the *set* event, like so:

```
tell application "TextWrangler" to set display font of text
window 1 to "Lucida Grande"
```

Let's go back to the *window* class for a moment. Most of the properties of this class are marked with the abbreviation *[r/o].* That stands for Read-Only. In other words, you can only *get* these properties, not *set* them.

Recordable Applications

Once an application accepts Apple Events, it actually makes a good deal of sense for an application to be designed in two parts: the user interface that you see, and the "engine" that does all the work. (An application designed this way is sometimes said to be *factored*.) The user interface then communicates with the engine via Apple Events.

The design of the Apple Event system makes it possible to "record" events into a script. This feature not only lets you automate frequently performed tasks with little hassle, it also can be an enormous aid in writing larger and more complicated scripts, because the application tells you what events and objects to use for the kind of task you record.

Because of the important recording functionality they enable, applications that have been factored and use Apple Events to let the two halves communicate are said to be *recordable.* It is important to note that not all scriptable applications are recordable.

Saving Scripts

Any AppleScript can be saved in what's called a *compiled script file*. A compiled script file contains the actual Apple Events; by generating these events when you save the file, the operating system does not have to convert your English-like commands into events each time you run the script, which means it loads faster. When double-clicked in the Finder, a compiled script file automatically opens in the Script Editor, where it can be run. A script can also be saved as a stand-alone application, or *applet*, in which case double-clicking the script's Finder icon automatically runs the script. Both types of files can be saved with or without the English-like *source code;* if you save it without the source code, other users you give the script to will not be able to make any changes to it (of course, you should also keep a copy of the script *with* the source for yourself).

Using Scripts with Applications

Although you can place a script applet in the global Scripts menu, or in any folder, and use it any time you need it, many applications (including TextWrangler) provide a special menu that lets you launch compiled scripts intended specifically for use with that one application. Since you do not have to save them as applets, they take up less disk space and launch more quickly. They also show up only in the application you use them with, rather than cluttering your global Scripts menu.

Some applications go even further, allowing you to define scripts to be run when certain things happen in the program. For example, an application might let you define a script to be executed when the user chooses *any* menu item. The script might then perform some pre-processing, and then exit by telling the application whether to continue with the menu command or to cancel it. As a simple example, a script might check to see what printer is selected when the user chooses the Print command. If it is the expensive color dye-sublimation printer, on which printing a page costs several dollars, the script could remind the user of that fact and confirm their intention (through an alert) before continuing with the print operation.

An application that supports such a feature (or any method of integrating user-written scripts seamlessly into its user interface) is said to be *attachable,* because the scripts become "attached" to the features of the program. (More details about using this feature are provided later in this chapter.)

Scripting Resources

Covering all the details you might need to write your own AppleScripts is not something we can reasonably do in this manual. AppleScript, despite its deceptively simple English-like syntax, is a sophisticated object-oriented language with many subtleties. For this reason, we suggest you consult supplemental documentation and resources if you are a beginning scripter.

A good place to start is with someone else's script: find a script that does *almost* what you want it to and repurpose it. Even if you cannot find a script that does anything close to what you want, reading others' scripts is a good way to learn how AppleScript "thinks" and how TextWrangler's particular AppleScript implementation behaves.

In addition to the basic AppleScript documentation included with the system, you may find the following resources useful in your quest to understand scripting.

Books

AppleScript: The Definitive Guide (Second Edition), Matt Neuberg. O'Reilly and Associates, 2006. ISBN: 0-596-10211-9

Discussion Groups

AppleScript Users

`http://www.lists.apple.com/applescript-users.html`
Official mailing list run by Apple for AppleScript users.

TextWrangler Talk

`http://groups.google.com/group/textwrangler`
The TextWrangler Talk discussion group is an excellent place to ask TextWrangler-specific scripting questions.

Mac Scripting

`http://listserv.dartmouth.edu/scripts/wa.exe?A0=MACSCRPT`
Unofficial list covers AppleScript and other Macintosh scripting languages, with occasional forays into peripheral topics.

Web Sites

AppleScript: The Language of Automation

`http://www.macosxautomation.com/applescript/`
An excellent starting point.

AppleScript — Apple Developer Connection

`http://developer.apple.com/AppleScript/`
Detailed information for developers and advanced users.

MacScripter.Net

`http://macscripter.net/`
A good selection of AppleScript-related news and topics, including the "AppleScript FAQ" and discussion forums.

ScriptWeb

`http://www.scriptweb.com/`

This site covers all scripting languages, not just AppleScript. Also, it has an extensive directory of scripting additions.

Software

Script Debugger

`http://www.latenightsw.com/`

Despite its name, Script Debugger is more than a debugger; it is actually an enhanced replacement for Apple's Script Editor, featuring variable monitoring, step/trace debugging, an object browser for an application's objects, and much more.

Using AppleScripts in TextWrangler

TextWrangler has been scriptable for years, and we have continually worked to refine its level of scripting support. In addition to providing extensive script access to its commands and data, TextWrangler is both attachable *and* recordable.

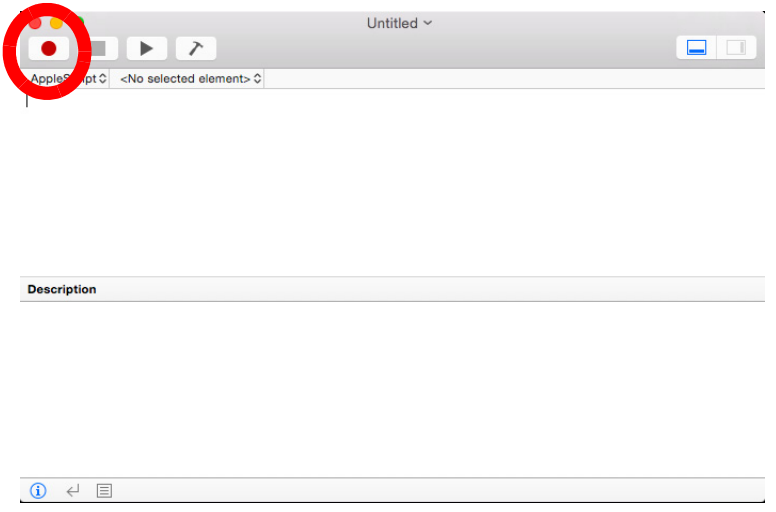
This section describes how you can create and employ AppleScripts within TextWrangler via recording and TextWrangler's various scripting facilities, while the following section covers TextWrangler's scripting commands and other issues related to preparing scripts for use.

Recording Actions within TextWrangler

Any language is easier to read than to write, easier to understand than to speak. AppleScript is no different. That's because, even though all the commands it uses are English words arranged in ways that more or less make grammatical sense, you still have to know (or find out from the application's dictionary) exactly which words to use, and what order they should go in. But it is easy to get started making scripts by recording them.

First, launch both TextWrangler and the Script Editor.

When you launch the Script Editor, a new, blank script window appears. Click the Record button, circled in the illustration below.



Now switch to TextWrangler and perform your task. Remember that the Script Editor is recording *everything* you do in every recordable application you are running, not just TextWrangler. If you do something in the Finder, for instance, that will get recorded too. Since almost everything you do is recorded, remember that if you make an error, and then Undo it, your recorded script will faithfully make the same mistake and undo it when you run it later. It will be possible to fix minor errors later, but things always go more smoothly if you do not make any mistakes, so take your time and try to do it right the first time.

Now switch back to the Script Editor and click the Stop button. After a brief pause, your script is compiled and ready for use. Try clicking the Run button to see it work. (It might not work correctly. If you recorded a search and replace operation changing every “cat” to “dog”, you already changed the document while recording the script, and of course the script will not do anything when you run it.)

Finally, save the script in the TextWrangler Scripts folder so that it shows up in TextWrangler’s script menu. Choose Save As from the File menu, and then use the Script Editor’s Save dialog to put the script in TextWrangler’s Scripts folder. Now try selecting it from the script menu in TextWrangler.

The Scripts Menu

The Scripts menu (left) in TextWrangler’s menu bar contains several commands. It also lists all AppleScripts (as well as text factories and Unix scripts) present in the Scripts folder within TextWrangler’s application support folder, providing a quick way to access frequently used scripts. You can place scripts within subfolders (up to 4 levels deep) of the Scripts folder to organize them.

Note AppleScripts written for use in as TextWrangler filters or scripts should be saved as compiled (data fork) script files, not script applications.

In addition to the list of available scripts, the Scripts menu provides the following commands.

Open Script Editor

Choose this item to switch to the system's default AppleScript editor. If the script editor is not running, TextWrangler launches it.

Open Scripting Dictionary

Choose this item to switch to your preferred AppleScript editor and open TextWrangler's scripting dictionary for viewing. If the script editor is not running, TextWrangler launches it.

Open Scripts Folder

Choose this item to open the Scripts folder which is located within TextWrangler's application support folder. (See "Scripts" on page 29.)

Running and Editing Scripts

Choose the item corresponding to any script to run that script. Hold down the Option key when choosing a script item to have TextWrangler open the script for editing in your preferred script editor, or hold down the Shift key when choosing a script item to have TextWrangler reveal the script file in the Finder. If you choose a folder node rather than a script item, TextWrangler will open the corresponding folder in the Finder.

The Scripts Palette

The Scripts command, located in the Palettes submenu of the Window menu, opens a palette listing all available scripts. Names that are too long to fit within the width of the window are truncated with ellipses (…).

"Hovering" the mouse over such a truncated name displays a tool tip showing the full name. If you hold down the Option key, the tool tip will appear instantly, with no hovering delay. Names that fit entirely within the window without truncation do not display a tool tip.

Organizing Scripts

Items in the Scripts menu or Scripts window are displayed in alphabetical order by default, but you can force them to appear in any desired order by including any two characters followed by a right parenthesis at the beginning of their name. (For example "00)Save All" would sort before "01)Close All.") For names of this form, the first three characters are not displayed in the window. You can also insert a divider by including an empty folder whose name ends with the string "-***". (The folder can be named anything, so it sorts where you want it.) These conventions are the same as those used by the utilities FinderPop and OtherMenu.

Attaching Scripts to Menu Items

TextWrangler lets you attach scripts to menu items. By this, we mean that you can write scripts that TextWrangler automatically calls before or after performing a menu command. For example, if you want TextWrangler's Open from FTP/SFTP Server command to launch your favorite FTP client, you can simply attach a script to that menu item. Scripts can return a value that tells TextWrangler whether to continue with the command that was selected, or to cancel the operation (in which case only the script is executed).

Scripts attached to TextWrangler menu items must be stored in the Menu Scripts folder of TextWrangler's application support folder. These files should be compiled scripts, not script applications. Scripts are named to indicate which menu item they go with: first the name of the menu (or the submenu) upon which the item is immediately located, then a bullet "•" (Option-8) character, then the name of the menu item. For example, to attach a script to the Open from FTP/SFTP Server menu item, you would name it "File•Open from FTP/SFTP Server", while to attach a script to the New Document menu item, you would name it "New•Text Document".

Two of TextWrangler's menus have icons rather than names. TextWrangler uses the following names for its icon menus: "#!" [the 'Shebang' menu], and "Scripts". Furthermore, the New With Stationery submenu is named "Stationery" for purposes of attachability.

When you choose a menu command which has an attached script, TextWrangler will pass the menu name and command (item) name to the script's MenuSelect handler, if it has one. If the script contains no MenuSelect handler, TextWrangler executes the script's run handler.

The script's MenuSelect handler can tell TextWrangler to skip performing the chosen command by returning "true", or have it continue on and perform the command by returning "false". If MenuSelect returns "false", TextWrangler will call the script's PostMenuSelect handler, if it has one, after it performs the menu command.

Here is a simple example, which adds a confirmation dialog to the Save command (addressed as “File•Save”). Note that we test the menu and item names to make sure the script is attached to the Save command—if it is attached to some other command, it does nothing.

```
on menuselect(menuName, itemName)
  if menuName = "File" and itemName = "Save" then
    set weHandledCommand to true
    display dialog "Are you sure you want to save?" ¬
    buttons {"No", "Save"} default button 2
    if button returned of the result is "Save" then
    -- the application should do its work
    set weHandledCommand to false
    else
    -- we handled the command, app does no work,
    -- postmenuselect doesn't get called
    display dialog "The document was not saved." ¬
      buttons {"OK"} default button 1
    end if
    return weHandledCommand
  end if
end menuselect

on postmenuselect(menuName, itemName)
  -- this is called after the application has processed
  -- the command
  display dialog "The document was saved." ¬
    buttons {"OK"} default button 1
end postmenuselect
```

Attaching Scripts to Events

IMPORTANT TextWrangler now offers enhanced script attachability: in addition to adding scripts to menu commands, you can now attach scripts to certain application and document events.

To access these events, your attachment scripts must contain function names which correspond to the names of the events’ attachment points. Except when otherwise noted, all of the following considerations apply:

- Every function takes a single argument which is a reference to the object in question: the application for application entry points, or the document being opened/closed/saved/etc for document entry points.
- Any function associated with an attachment point whose name contains ‘should’ must return a Boolean result: ‘true’ or ‘false’. If it returns ‘true’, the operation will continue. If it returns ‘false’ or throws an error (see below) then the operation will be cancelled. So, for example, ‘applicationShouldQuit’ returning ‘true’ will allow the application to quit; returning ‘false’ will not.
- If an attachment script causes a scripting error and does not handle it within the script itself, TextWrangler will report the error. In the case of functions which are used to allow a ‘should’ action, this will prevent the action from occurring.

Here are the available attachment points:

Application attachment points

- *applicationDidFinishLaunching*: called when the application has completed startup.
- *applicationShouldQuit*: called when you choose the Quit (or the application receives a 'quit' event for any other reason).
- *applicationDidQuit*: called when the application has finished shutting down and is about to exit.

Document attachment points

- *documentDidOpen*: called when a document has been opened and is ready for use. (Since TextWrangler supports multiple types of documents, your script should allow for the argument to be a document of any type.)
- *documentShouldClose*: called when the application is preparing to close a document.
- *documentDidClose*: called when the application has closed a document.
-
- *documentShouldSave*: called when the application is trying to determine whether a given document should be saved.
- *documentWillSave*: called when the application is about to begin saving a document. (note that this will only be called after a successful return from a 'documentShouldSave'.
- *documentDidSave*: called after a document has been saved successfully.
- *documentWillUnlock*: called when TextWrangler is going to make a document writeable. (For example, when you click the pencil to unlock a document)
- *documentDidUnlock*: called when TextWrangler has successfully made a document writeable.
- *documentWillLock*: called when TextWrangler is going to make a document read-only.
- *documentDidLock*: called when TextWrangler has successfully made a document read-only.

Using Attachment Scripts

Scripts attached to events must be stored in the "Attachment Scripts" folder of TextWrangler's application support folder (see page 28).

You can write one script to handle each attachment point, or one script to handle the attachment points for an entire class of objects, or one script to handle all of the attachment points for the entire application.

You can also mix and match scripts to meet specialized needs: for instance, by using one script to implement a particular attachment point for documents, and a second script to handle the remaining attachment points.

TextWrangler associates scripts to attachment points by means of the script's file name. There are three ways to specify a script's role:

1. **<ObjectClass>.<entryPoint>**
2. **<ObjectClass>**
3. **<ApplicationName>**

The first form is the most specific: the 'ObjectClass' may be either "Application" or "Document", while the 'entryPoint' is one of the attachment points described above appropriate to that object class.

For example, a script which implemented only the *documentDidSave* attachment point should have the name "Document.documentDidSave.scpt" and contain a subroutine named 'documentDidSave', thus:

```
on documentDidSave

    -- do something useful and appropriate

end documentDidSave
```

Note Adding the filename suffix '.scpt' is not mandatory, but you should follow the current system conventions suggested when creating scripts with the AppleScript Editor (or any other script editor such as Script Debugger).

The second form allows you to implement all of the attachment points for a single object class in a single script file, if desired.

For example, you could create a script named "Application.scpt" containing subroutines for as many of the application attachment points as you wish:

```
on applicationDidFinishLaunching

    -- do something relevant

end applicationDidFinishLaunching

on applicationShouldQuit

    -- hello world

    return (current date as string contains "day")

end applicationShouldQuit
```

Likewise, to implement all of the attachment points for the Document class, you could create a script named “Document.scpt”, and put subroutines in it for the document attachment points:

```
on documentDidSave

    -- do something relevant

end documentDidSave

    ...

on documentWillClose

    ...

end documentWillClose
```

The third form allows you to write a single all-encompassing script which contains subroutines for all of the attachment points in the application. To do this, name the script “TextWrangler.scpt” and include whatever subroutines you wish to implement. For example:

```
on applicationShouldQuit

    -- hello world

    return (current date as string contains “day”)

end applicationShouldQuit

on documentWillClose

    ...

end documentWillClose
```

When figuring out which script to run, TextWrangler will first look for a script whose name exactly matches the attachment point, e.g. “Document.documentShouldSave.scpt”. If there is no such script, TextWrangler will then look for a script whose name matches the object class at the attachment point, e.g. “Document.scpt”. Finally, if there are no scripts with either an exact or a class match, TextWrangler will look for an application-wide script: “TextWrangler.scpt”.

Note You do **not** have to implement attachment subroutines for all attachment points, or for all classes—only the ones you need. If there is no attachment script or subroutine, TextWrangler proceeds normally.

Using an Attachment Script to Perform Authenticated Saves

TextWrangler 4.0 and later support a special attachment point for the Document class: *documentShouldFinalizeAuthenticatedSave*. This attachment point will be called whenever an authenticated save is necessary (for text documents only).

The following sample script illustrates how to use this facility (the comments are important, so please read them!):

```
on documentShouldFinalizeAuthenticatedSave(theDocument,
tempFilePath, destinationPath)

    -- on input: tempFilePath points to the contents
    -- of the document written to a temp file, ready
    -- to move to the destination; destinationPath is
    -- where the file should be copied.

    -- on exit: if the operation succeeded, delete the
    -- temp file (or else the application will assume
    -- the operation failed) and return YES for success

    -- this is pretty straightforward:
    --   "cp tmpFilePath destinationPath"

   do shell script "cp" & " " & quoted form of tempFilePath
& " " & quoted form of destinationPath with administrator
privileges

    -- now remove the temp file, this indicates to
    -- the application that we did the work

    do shell script "rm" & " " & quoted form of tempFilePath

    return true

end documentShouldFinalizeAuthenticatedSave
```

Filtering Text with AppleScripts

The Text Filters folder in TextWrangler's application support folder contains executable items in the form of compiled AppleScripts and Unix filters, which you may apply to the active document via the Apply Text Filter command in the Text menu.

When you apply such an item, TextWrangler will pass either the selected text (or the contents of the active document, if there is no selection) as a reference to a 'RunFromTextWrangler' entry point within your AppleScript, and your script should return a string which TextWrangler will use to replace the selected text (or the contents of the document). If your script does not contain a 'RunFromTextWrangler' entry point, TextWrangler will call its run handler, again passing a reference to the current selection range.

TextWrangler's Scripting Model

This section provides a high-level overview of TextWrangler's scripting model that will, where appropriate, contrast the current scripting framework against older versions of TextWrangler, and suggest how you can modify your existing scripts for compatibility.

IMPORTANT Because TextWrangler's scripting dictionary changes whenever we add features, it should be considered the definitive reference in any situation where it and this document differ. We have found Script Debugger from Late Night Software to be an excellent tool for browsing and navigating TextWrangler's scripting dictionary, as well as for preparing and testing scripts.

```
http://www.latenightsw.com/
```

Script Compatibility

Scripts prepared for older versions may need to be revised in order to work properly. Further, since all recent versions of TextWrangler allow multiple documents to be open within a single text window, you may need to revise other existing scripts.

Distinguishing Between Script Elements

Because different applications handle different types of data, you should be aware that the actual data, or the interface items, referred to by a particular name may not be consistent from application to application. The following sections describe how several common elements are handled in TextWrangler.

Applying Commands to Text

Since TextWrangler supports opening multiple documents within a single text window, all scripting commands which operate on text must specifically target the text contents of a window, or a document within that window, rather than the window itself.

For example, you may use:

```
count lines of text of document of text window 1
```

or:

```
count lines of active document of text window 1
```

but not:

```
count lines of text window 1
```

Documents vs. Windows

In old versions of TextWrangler, the object classes *document* and *window* could be used interchangeably, and generally had the same properties listed in the scripting dictionary. This is no longer the case.

The class *window* now corresponds to a window (of any type—text or otherwise) on screen, and thus the properties of the *window* class now refer strictly to properties of a window on screen. If a document is associated with a window, the document is accessed as the *document* property of the window:

```
document of text window 1
```

The class *document* refers to a document, and as with a window, the document's properties pertain strictly to the condition of a document (that is, something that can be saved to disk and opened later). Note that this does not mean a document must be saved to a file, only that it could be.

As a rule, documents and windows are associated with each other, but it is important to remember that there is not a one-to-one correspondence between windows and documents. For example, the About box is a window which has no document associated with it. Furthermore, in current versions of the application, there is no such thing as a document with no associated window.

Here is a general overview of the object classes used in TextWrangler:

Classes of Windows

- *window*: the basic window class contains properties that can be fetched and set for any window on screen: position, size, and so forth.
- *palette*: the palette class refers to windows that float above all others on the screen; the Character Inspector, Scripts, and Text Filters palettes, and so on.
- *text window*: the text window class provides properties which are specific to text-editing windows as on-screen entities. These properties pertain mostly to the display of text in the window: *show invisibles*, *auto_indent*, and so on. In addition to the text-editing-specific properties, the basic window properties are also accessible.
- *disk browser window*: provides a way to reference windows corresponding to open disk browsers. A disk browser window does not present any properties beyond the basic *window* class, but provides a way to differentiate disk browser windows from other types of window.
- *results browser*: provides a way to reference results generated by a batch operation. A results browser does not present any properties beyond the basic *window* class, but provides a way to differentiate results windows from other types of window.
- *search results browser*: a subclass of results browser, referringspecifically to the results of a single-file Find All command or a multi-file search.

Classes of Document

As with windows, there are various classes of document:

- *document*: the basic document class contains properties that apply to any sort of document: whether it has unsaved changes, the alias to the file on disk, and so on.
- *text document*: text documents contain information specific to text files opened for editing in TextWrangler.
- *picture document*: refers to a document corresponding to an open picture file. A picture document does not present any properties beyond the basic *document* class, but provides a way to differentiate picture documents from other types of document.

- *movie document*: refers to a document corresponding to an open QuickTime movie file. A movie document does not present any properties beyond the basic "document" class, but provides a way to differentiate movie documents from other types of document.
- *QuickTime document*: refers to a document corresponding to an imported Quicktime image file. A QuickTime document does not present any properties beyond the basic "document" class, but provides a way to differentiate QuickTime documents from other types of documents.

"Lines" and "Display_lines"

The "line" element refers to a "hard" line, that is, a stream of characters that begins at the start of file or after a line break, and which ends at the end of file or immediately before a line break. This is consistent with the previous semantics of "line" in hard-wrapped documents, and these semantics now apply in soft-wrapped documents as well.

The "display_line" element refers to a line of text as displayed on screen (bounded by soft and/or hard line breaks).

The "startLine" and "endLine" properties of a text object now always refer to the "hard" start and end of lines. In other words, if a text object crosses multiple soft-wrapped lines, the startLine and endLine properties will be the same.

Both "startDisplayLine" and "endDisplayLine" properties are now part of the text object class. These serve the same purpose as the startLine and endLine semantics for soft-wrapped views in older versions of TextWrangler.

Getting and Setting Properties

A significant feature of TextWrangler's scripting framework is the ability to get and set multiple properties of an object with a single scripting command. Every object has a property called *properties*. This property returns a record which contains all of the properties which can be fetched for that object. For example, the script command

```
properties of text window 1
```

will return a result like this one:

```
{{id:55632400, container:application "TextWrangler", bounds:{31,
44, 543, 964}, closeable:true, collapsed:false, index:1,
modal:false, file:alias "Hard
Disk:Users:Shared:doc_examples:index copy.html", modified:false,
name:"index copy.html", position:{31, 44}, resizable:true,
selection:"", contents:"..."}
```

Conversely, to set one or more properties at once is very easy:

```
set properties of text window 1 to { show invisibles: true, show
spaces : true, soft wrap text : true }
```

Only the properties specified will be changed. The rest will not be modified.

It is important to note that when setting properties in this fashion, you can only set modifiable properties. If you attempt to set any read-only properties, a scripting error will result:

```
set properties of text window 1 to { show invisibles: true,
modal: false, expand tabs: true }
```

The above script command will turn on Show Invisibles and then report a scripting error, since *modal* is a read-only property.

Performing Actions

The following sections provide basic information on how to perform various common actions via AppleScript.

Scripting Searches

The ability to script searches presents you with a very powerful tool, since you can prepare a script which instructs TextWrangler to perform a whole series of search or search and replace operations.

Consider the scripting command below:

```
tell application "TextWrangler"

find "TextWrangler(.+)$" searching in document of text window 1
¬
  options { search mode: Grep } with selecting match

end tell
```

In previous versions, the *find* command always operated on the front window. Now, you must explicitly specify the text to be searched, either by specifying an explicit tell target, or by supplying a *searching in* parameter. So the following scripts are equivalent:

```
tell application "TextWrangler"
    find "TextWrangler" searching in document of text window 1
end tell
```

and

```
tell application "TextWrangler"
    tell document of text window 1
        find "TextWrangler"
    end tell
end tell
```

Note that either the tell-target or the *searching in* parameter must resolve to something that contains text. As a shortcut, you can specify a window, and if the window contains text, the search can proceed. You can also specify a text object:

```
find "Search Text" searching in (lines 3 thru 5 of document of
text window 2)
```

Also unlike previous versions of TextWrangler, the defaults for parameters not specified in the *find* command are no longer controlled by the user interface (that is, the Find window).

When performing a *find*, TextWrangler will return a record describing the results of the search. This record contains a Boolean which indicates whether the search was successful, a reference to the text matched by the search, and the text string matched by the search. Given the first example above, the results might look like this (after reformatting for clarity):

```
{found:true,
found object:characters 55 thru 60 of text window 1 of
application "TextWrangler",
found text:"TextWrangler"}
```

Scripting Single Replaces

To do a single find and replace via AppleScript, you can write:

```
tell application "TextWrangler"

set result to (find "TextWrangler" searching in text window 1¬
    with selecting match)

    if (found of result) then
       set text of (found object of result) to "Replacement"
    end if

end tell
```

When performing a grep search, you cannot just replace the matched pattern with a replacement string; the grep subsystem needs to compute the substitutions. The *grep substitution* event is provided for this purpose; given a preceding successful Grep search, it will return the appropriate replacement string. So if you perform a grep search, the script would look like:

```
tell application "TextWrangler"

set result to find "TextWrangler(.+)$" searching in text window
1 ¬
     options {search mode:grep}

    if (found of result) then
         set text of (found object of result) to ¬
             grep substitution of "\\1"
    end if

end tell
```

Note that when using a backslash "\" character in AppleScript, it needs to be "escaped" by means of another backslash; thus, in the above example, "\\1" used in the script, will become the grep replacement string "\1" when passed to TextWrangler.

Scripting Multi-File Searches

In TextWrangler, a multi-file search is a simple extension of the *find* scripting command. To search a single file or folder for all occurrences matching the search parameters, specify the file or folder as the *searching in* parameter of the search.

For example, to find all occurrences of "index.html" in a web site, one might use the following scripting command:

```
find "index.html" searching in (alias "Files:WebSite:")
```

Likewise, to find JavaScript line comments:

```
find "//.+$" searching in (alias "Files:WebSite:") ¬
    options {search mode: Grep}
```

To search in a single file:

```
find "crash" searching in (alias "Files:WebSite:index.html")
```

Scripting the Clipboard

TextWrangler has multiple clipboards. These are fully accessible via the scripting interface. Due to operating system constraints, most clipboard operations require TextWrangler to be frontmost.

Here are some examples:

```
count clipboard
```

- Returns the number of clipboards supported by the application

```
clipboard 1
```

- Returns {index:1, contents:"Files:WebSite:", length:14, is multibyte:false, display font:"ProFont", display font size:9, style:{plain}}

```
clipboard 1 as text
```

- Returns "Files:WebSite:"

```
clipboard 1 as reference
```

- Returns clipboard 1 of application "TextWrangler"

```
current clipboard
```

- Returns the current clipboard as a record (you can coerce it to reference or text or get individual properties)

To set the text in a given clipboard to literal text:

```
set contents of clipboard 3 to "foobar"
```

To set the text in a clipboard to text represented by an object specifier:

```
set contents of clipboard 3 to selection of window 2
```

To copy the contents of one clipboard to another:

```
set contents of clipboard 5 to clipboard 3
```

or, to set the current clipboard to the contents of a different clipboard, (thus making it exportable to the system clipboard):

```
set current clipboard to clipboard 3 as text
```

or finally, with even less typing involved:

```
set current clipboard to clipboard 5
```

To make any clipboard the current clipboard, select it:

```
select clipboard 5
```

Setting Text Encodings

When specifying the encoding to use for opening or saving a file, you may either use the encoding's internet name, or its exact display name (as shown in the Read As popup menu).

For example:

```
open {file "Hard Disk:Users:Shared:example.txt"} reading as
"Western (ISO Latin 1)"
```

```
open {file "Hard Disk:Users:Shared:example.txt"} reading as
"iso-8859-1"
```

Arranging Documents and Windows

TextWrangler provides considerable control for handling windows and documents both directly and via AppleScript.

Opening Documents

The "open" command supports additional options, which allow you to override your window handling preferences on a case by case basis:

```
open aFileList opening in <value>
```

As in previous releases, <value> may be a reference to an existing text window. However, you may instead specify "front_window", "new_window", or "separate_windows", which have the following effect:

- front_window: All files in aFileList are opened in the frontmost text window. (If there is no text window open, TextWrangler will create a new one.)
- new_window: All files in aFileList are opened into a new text window.
- separate_windows: Each file in aFileList is opened into its own text window.

Moving Documents

The “move” command can be used to move text documents between text windows. For example:

```
tell application "TextWrangler"
    if (count of text windows) > 0 then
        select text window 1
        repeat while (count of text windows) > 1
            set ct to count documents of text window 2
            repeat with i from 1 to ct
                move document 1 of text window 2 to text window 1
            end repeat
        end repeat
    else
        beep
    end if
end tell
```

Referencing Documents

Previously, documents were indexed inside of multi-document windows by their display order in the file list. This meant that “document 1” of the application might not be the active document, which in turn required scripts to make special provisions to deal with the presence of multiple documents in a single window.

In order to handle this, TextWrangler 2.0 provided the “active document” property, which you could always use to specify the currently active document of a given text window. For example:

```
active document of text window 1 of application “TextWrangler”
```

Although TextWrangler still supports the “active document” property, this is no longer necessary. Instead, if a text window is frontmost:

```
document 1 of application "TextWrangler"
```

```
document 1 of text window 1 of application "TextWrangler"
```

```
active document of text window 1 of application "TextWrangler"
```

now all refer to the same document. The side effect of this change is that if you wish to access documents within a text window by index, that index is:

a) not related to the visual ordering of documents in the file list, and,

b) documents’ indexes may change over time

This situation is effectively no different than handling documents which are contained in individual text windows, i.e. the index will change over time when you select different windows. If your script needs to keep a permanent references to a particular document, you should refer to that document by its id rather than its index.

Common AppleScript Pitfalls

Here are some things to watch out for when scripting TextWrangler with AppleScript.

The Escape Issue

AppleScript uses the backslash character as an escape character. You can use \r to indicate a carriage return or \t to indicate a tab character. More importantly, you can use \" or \' to include a quote mark or apostrophe in a string that is delimited by quotes or apostrophes. If you want to specify a literal backslash, you must write \\ i.e. a pair of backslashes.

That's not all that confusing until you start writing AppleScripts that call on TextWrangler's powerful grep searching capability. TextWrangler *also* uses the backslash as an escape character. If you want to search for an actual backslash in a document, you have to tell TextWrangler to search for \\. However, if you do that in AppleScript, you must keep in mind that AppleScript will first interpret the backslashes before passing them to TextWrangler. To pass one backslash to TextWrangler from AppleScript, you must write two in AppleScript.

So to tell TextWrangler to search for a single literal backslash from an AppleScript, you must write no fewer than *four* backslashes in the script. Each pair of backslashes is interpreted as a single backslash by AppleScript, which then passes two backslashes to TextWrangler. And TextWrangler interprets those two backslashes as a single one for search purposes. (This proliferation of backslashes can make your scripts look a bit like a blown-over picket fence.)

The Every Item Issue

When writing a script that loops through every item of a TextWrangler object (for example, every line of a document), do not do it like this:

```
repeat with i in every line of text document 1
    -- do stuff here...
end repeat
```

This forces TextWrangler to evaluate "every line of document 1" every time through the loop, which will slow your script significantly. Instead, write

```
set theLines to every line of text document 1
repeat with i in theLines
    -- do stuff here...
end repeat
```

CHAPTER

12 Unix Scripting and the Command-Line

This chapter describes how to set up TextWrangler to work with development environments. TextWrangler offers multiple features which support development tasks, beginning with syntax coloring and function browsing support for numerous languages, and continuing to direct integration with the system-supplied Perl, Python, and Ruby environments, as well as shell scripts and other Unix scripting tools. Additionally, you can invoke TextWrangler from the command line via its optional command line tools.

In this chapter

Configuring TextWrangler **225**
Syntax Coloring – 225 • *Switching Between Counterpart Files* – 226
TextWrangler and the Unix Command Line **226**
Installing the Command Line Tools – 226
The "edit" Command Line Tool – 226
The "twdiff" Command Line Tool – 227
The "twfind" Command Line Tool – 227
Unix Scripting: Perl, Python, Ruby, Shells, and more! **229**
Using Unix Scripts – 229 • *Language Resources* – 229
Setting Environment Variables for GUI Apps – 230
Configuring Perl – 231 • *Configuring Python* – 231
Configuring Ruby – 231 • *Shebang Menu* – 231
Filters and Scripts – 232 • *Filters* – 233 • *Scripts* – 234
Additional Notes – 234

Configuring TextWrangler

The Shebang (#!) menu is always available by default to provide you access to TextWrangler's support for running Unix scripts.

Syntax Coloring

Syntax coloring is the practice of drawing keywords and other language elements in colors which differ from the standard text color to add emphasis and improve the readability of your code. TextWrangler offers built-in syntax coloring support for a wide range of programming languages and other types of structured content. You can adjust TextWrangler's default text colors or define color schemes in the Text Colors preference panel, or assign a color scheme to a specific language in the Languages preference panel.

Switching Between Counterpart Files

When editing any source file which has a counterpart (header), you can press the Counterpart button in the navigation bar or type Control-Option-up arrow to switch to its counterpart file, or vice versa. (TextWrangler uses the suffix mapping options in the Languages preference panel to determine whether a particular file is a source or header file.)

TextWrangler and the Unix Command Line

This section describes TextWrangler's facilities for interacting with the Unix command line: shell worksheets for issuing commands *to* the Unix shell and the "edit", "twdiff", and "twfind" command line tools for invoking TextWrangler *from* the command line.

Installing the Command Line Tools

Though TextWrangler will not automatically attempt to install its command line tools, you may do so at any time by choosing "Install Command Line Tools" from the TextWrangler (application) menu at any time.

If older versions of the tools are installed, choosing this command will update them; it will not overwrite existing versions of the tools with older versions.

The "edit" Command Line Tool

You can use the "edit" command line tool to open files into TextWrangler via the Unix command line.

To open a file into TextWrangler from the command line, type

```
edit filename
```

where *filename* is the name of the file to be opened. You may also specify a complete FTP or SFTP URL to a remote file or folder to have TextWrangler open the file, or an FTP/SFTP browser to the folder.

To launch TextWrangler without opening a file (or to activate the application if it is already running), type

```
edit -l
```

You can also pipe STDIN to the "edit" tool, and it will open in a new untitled window in TextWrangler: for example,

```
ls -la | edit
```

If you just type

```
edit
```

with no parameters, the tool will accept STDIN from the terminal; type Control-D (end-of-file) to terminate and send it to TextWrangler.

The complete command line syntax for the “edit” tool is

```
edit [ -bcChlpsuvVw --resume ] [ -e <encoding_name> ]
    [ -t <string> ] [ +<n> ] [ file (or) <S/FTP URL> ... ]
```

See the “edit” tool’s man page (“man edit”) for a complete description of the available switches and options.

The “twdiff” Command Line Tool

You can use the “twdiff” command line tool to apply TextWrangler’s Find Differences command to a pair of files or folders specified on the Unix command line.

To invoke the Find Differences command from the command line, type

```
twdiff oldfile newfile
```

or

```
twdiff oldfolder newfolder
```

where *oldfile* and *newfile* are the names of the files, or *oldfolder* and *newfolder* are the names of the folders, to be compared. You can also specify options for how the Find Differences command will be applied, which correspond to those available in the dialog.

The complete command line syntax for the “twdiff” tool is

```
twdiff [ --<options> ] [ OLDFILE NEWFILE | OLDFOLDER NEWFOLDER
 ]
```

See the “twdiff” tool’s man page (“man twdiff”) for a complete description of the available switches and options.

Invoking “twdiff” as an External Helper

When using “twdiff” as an external diff helper for any other program, e.g. Subversion, you should invoke it with the --wait option.

The “twfind” Command Line Tool

You can use the “twfind” command line tool to access TextWrangler’s powerful multi-file search from the Unix command line.

To perform a multi-file search from the command line, type

```
twfind search-string search-path
```

where *search-string* is your search string (or pattern) and *search-path* is a list of path(s) to search. You can also specify options which control how the search should be performed; these options correspond to those available in the Multi-File Search window.

If no search paths are specified on the command line, “twfind” will attempt to read them from standard input. This makes it easy to process the output of other tools such as “find”. For example:

```
`find . -name "*.py" -print | twfind blah`
```

takes the paths printed by “find” and searches those files.

By default, “twfind” expects that input will be separated by Unix newlines (\n). If instead, the input is being generated programmatically and contains “NUL”-separated paths, you can specify the “-0” option. Again using “find” as an example input source:

```
`find . -name "*.py" -print0 | twfind blah -0`
```

The complete command line syntax for the “twfind” tool is

```
twfind search-string [-cEghInRSvVwZ0 --<long_form_switches>
  search-path ]
```

See the “twfind” tool’s man page (“man twfind”) for a complete description of the available switches and options.

Unix Scripting: Perl, Python, Ruby, Shells, and more!

TextWrangler provides robust integration with numerous Unix scripting environments, including Perl, Python, Ruby, and shell scripts.

Using Unix Scripts

TextWrangler works directly with the native Perl, Python, and Ruby environments provided by Mac OS X, and supports similar integration with shell scripts and any other Unix scripting language.

TextWrangler's Unix scripting features are accessed via the Shebang menu: "#!". (Why "Shebang"? Because executable Unix scripts traditionally start with the two-character sequence "#!". Some people pronounce these two characters "hash-bang," others say "sharp-bang," but the most common pronunciation is simply "shebang.")

The "shebang line" is the first line of the script, and includes a Unix-style path to the interpreter for the language—for example, "#!/usr/bin/perl", or "#!/usr/local/bin/python".

While TextWrangler does not entirely depend upon the accuracy of the shebang line (if your script file has an accurate language mapping), it is always a good practice, and sometimes necessary, to specify the full path to the executable in the shebang line.

Language Resources

Perl is an acronym for Practical Extraction and Report Language (or alternatively, Pathologically Eclectic Rubbish Lister) and was developed by Larry Wall. If you are interested in learning Perl, the quintessential Perl references are:

Learning Perl (4th Edition), by Randal L. Schwartz & Tom Phoenix. O'Reilly and Associates, 2005. ISBN: 0-596-10105-8

Programming Perl (3rd Edition), by Larry Wall, Tom Christiansen, Jon Orwant. O'Reilly and Associates, 2000. ISBN: 0-596-00027-8

The following are excellent Internet resources for the Macintosh implementation of Perl, and Perl in general:

Perl.com from O'Reilly and Associates
http://www.perl.com/

Perl Mailing Lists
http://lists.cpan.org/

Python is a portable, interpreted, object-oriented programming language, originally developed by Guido van Rossum. If you are interested in learning Python, consider the following Internet resources:

Python home page
`http://www.python.org`

Python Cookbook
`http://aspn.activestate.com/ASPN/Cookbook/Python`

Ruby is an interpreted scripting language with an emphasis on object-oriented programming, which has fast become a favorite of Web developers. Ruby was created by Yukihiro Matsumoto. If you are interested in learning Ruby, consider the following books:

Programming Ruby: The Pragmatic Programmer's Guide (2nd Edition), by Dave Thomas, with Chad Fowler and Andy Hunt. Pragmatic Bookshelf, 2004.
ISBN: 0-9745140-5-5

Ruby Cookbook, by Lucas Carlson & Leonard Richardson. O'Reilly and Associates, 2006.
ISBN: 0-596-52369-6

Internet resources for Ruby:

Ruby home page
`http://www.ruby-lang.org/`

RubyGarden Wiki
`http://wiki.rubygarden.org/Ruby`

Setting Environment Variables for GUI Apps

TextWrangler now reads your account's command line environment directly; thus, you need not employ any special mechanisms to pass environment settings to it.

Line Endings, Permissions and Unix Scripts

To execute scripts, the script interpreter for any given language requires source code to be encoded with native line endings, i.e. Unix line breaks for Perl and most other shell scripting languages. TextWrangler will warn you if you attempt to run a script which does not have Unix line endings.

Additionally, to execute scripts anywhere outside of TextWrangler (e.g. in the Terminal), the system requires that the script file have 'execute' permissions set. Thus, when you first save any script file which contains a shebang (#!) line, TextWrangler will automatically set execute permissions for your login account (a+x, as modified by the umask) on that file.

Configuring Perl

TextWrangler can make full use of the system's default Perl install with no need for further configuration. However, if you wish to install and work with multiple versions of Perl, you will need to specify the appropriate version in your scripts' shebang lines.

Search Paths

By default, Perl looks for modules in its standard library path and in the current directory. You may also use modules from other locations by specifying their paths in the PERL5LIB environment variable.

To do so, you must first create an "environment.plist" file as described under "Setting Environment Variables for GUI Apps" on page 230, then add a key-value pair consisting of the variable and the desired paths as follows:

```
<key>PERL5LIB</key>
<string>/Users/example/Sandbox/myPerlMods/:/Users/Shared/
moreMods/</string>
```

(You may specify multiple directories by separating their paths with colons per the standard convention.)

Configuring Python

TextWrangler expects to find Python in /usr/bin, /usr/local/bin, or /sw/bin. If you have installed Python elsewhere, you must create a symbolic link in /usr/local/bin pointing to your copy of Python in order to use pydoc and the Python debugger.

Configuring Ruby

TextWrangler can make full use of the system's default Ruby install with no need for further configuration. However, if you wish to install and work with multiple versions of Ruby, you will need to specify the appropriate version in your scripts' shebang lines.

Shebang Menu

The commands in this menu allow you to run Unix scripts directly within TextWrangler.

Check Syntax

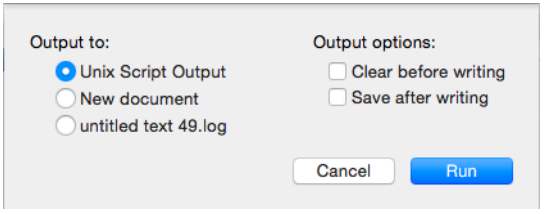
Checks the syntax for the frontmost window. Errors are displayed in a standard TextWrangler error browser (see Chapter 9, "Browsers," for more details on working with error browsers). This command is only available for Perl and Python scripts.

Run

Runs the script in the frontmost window by default. If this script has an associated disk file, TextWrangler will automatically set the current working directory to the directory containing the script file, and any output from the script (on STDOUT) will be displayed in TextWrangler's "Unix Script Output" window. By default, errors for Perl and Python scripts are displayed in an error browser, while errors for other languages will be written into a new document.

Run with Options...

Displays the Run a Script sheet, which allows you to set options before running the script in the frontmost window.



Output to: Choose to display output in a new document, to direct it to the Unix Output file, or to write it to an equivalently-named log file ("{script name}.log") in TextWrangler's Logs folder (~/Library/Logs/TextWrangler/).

Output Options: Mark these checkboxes to clear the output file before writing and to save it after writing, respectively.

Run in Terminal

This command will run the script in a new Terminal window, regardless of the settings in the Run a Script dialog.

Run in Debugger

Runs the script in the interpreter's debugger, regardless of whether the Use Debugger option is set for the Run command; also, any output options set in the Run command will be ignored. The Run in Debugger command is only available for Perl and Python.

Run File

Runs a script from an arbitrary file rather than from a TextWrangler window. The Run a Script File dialog appears. You can select a file by clicking the File button or by dragging a file to the path box at the top of the dialog from the Finder. The options are the same as the ones described above for the Run a Script dialog.

Show POD/Show Module Documentation

When the frontmost document is a Perl file and you invoke the Show POD command, TextWrangler will process the document contents using by the command line pod2text tool and display the result in a new text window.

Note POD stands for Plain Old Documentation, and is the standard Perl documentation format.

When the frontmost document is a Python file, the name of this command will change to Show Module Documentation, and if you invoke it, TextWrangler will display the module documentation.

Filters and Scripts

Before you begin using Unix filters and scripts with TextWrangler, you should locate and familiarize yourself with the Text Filters and Scripts folders, which resides within TextWrangler's application support folder. (See Chapter 2 for details.).

The contents of the Text Filters and Scripts subfolders are presented respectively in the Apply Text Filters submenu and the Scripts menu, as well as the Text Filters and Scripts floating palettes.

Document State

For convenience, TextWrangler sets some runtime environment variables to provide information about the front document's state right before a Unix filter or script is run:

Variable	Description
BB_DOC_LANGUAGE	Name of the document's current language (not set if language is "none")
BB_DOC_MODE	Emacs mode of the document's current language
BB_DOC_NAME	name of the document
BB_DOC_PATH	path of the document (not set if the document is unsaved)
BB_DOC_SELEND	(zero-based) end of the selection range (not set if not text document)
BB_DOC_SELEND_COLUMN	(one-based) de-tabbed column number of BB_DOC_SELEND
BB_DOC_SELEND_LINE	(one-based) line number of BB_DOC_SELEND
BB_DOC_SELSTART	(zero-based) start of the selection range (not set if not text document)
BB_DOC_SELSTART_COLUMN	(one-based) de-tabbed column number of BB_DOC_SELSTART
BB_DOC_SELSTART_LINE	(one-based) line number of BB_DOC_SELSTART

Note Selection ranges and other offsets are expressed in **characters,** not bytes.

Filters

Filters operate on the selected text of the frontmost document. TextWrangler will pass either the selected text (if any) or the contents of the entire document as input to the filter on STDIN and any output generated by the filter overwrites the selection.

Note This method represents a change from versions 3.5.3 and prior, in which TextWrangler wrote a temporary file and passed it on `argv[0]`. Thus, if you have any existing Unix filters (in the "Text Filters" folder), you **will** need to modify those filters in order for them to continue working.

There are two ways to run filters: through the Apply Text Filters submenu in the Text menu or via the Text Filters palette. To open the Text Filters palette, select it from the Palettes submenu in the Window menu. You can run a filter by selecting it from the list and clicking the Run button, or you can simply double-click the filter name in the list.

Hold down the Option key while double-clicking a filter or selecting it from the menu to open the file for editing instead of running it. You can also hold down the Shift key while selecting a filter item from the Apply Text Scripts submenu to reveal the file in the Finder, or you can select a folder node from the menu to open that folder in the Finder.

Scripts

Scripts do not operate on the text of the frontmost window, but rather run directly. You can run scripts from the Scripts menu or the Scripts palette. Hold down the Command key while selecting or double-clicking a script to open the Run a Script options dialog; hold down the Option key while selecting or double-clicking a script to open the script's file for editing instead of running it; hold down the Shift key while selecting or double-clicking a script to reveal its file in the Finder (or while selecting a folder node, to reveal that node in the Finder).

Additional Notes

In addition to the features detailed above, TextWrangler offers some additional options which it may help you to be aware of.

Setting Menu Keys for Scripts

The Filters and Scripts palettes both have a "Set Shortcut" button at the top. Select a filter or script in the list and click this button to set a keyboard shortcut for the selected item. You may also assign key equivalents to scripts or filters within the Menus & Shortcuts preference panel.

Manually Sorting the Filter and Script Lists

By default, items in the Apply Text Filters submenu and the Scripts menu display in alphabetical order. However, you can force items to appear in any desired order by including any two characters followed by a right parenthesis at the beginning of their name. (For example "00)Foo" would sort before "01)Bar.") For such files, the first three characters are not displayed in TextWrangler. You can also insert a divider by including an empty folder whose name ends with the string "-***". (The folder can be named anything, so it sorts where you want it.)

Canceling Filter or Script Execution

You can press the Cancel button in the progress dialog or type Command-. (Command-period) to cancel a task directly from within TextWrangler. Since TextWrangler must kill the spawned Unix process with a SIGINT, any unflushed data in open filehandles (including STDOUT and STDERR) will be lost unless the script takes measures to prevent this.

CHAPTER

13 Language Modules

Language modules are special files that you can install to add support for syntax coloring, and optionally, function browsing, for programming languages beyond those built in. Many people have prepared language modules for use with BBEdit and TextWrangler, and these modules are available from various web sites (including our own).

This chapter describes the basic procedures for installing and using language modules, and provides references to information about producing such items.

In this chapter

Language Modules 235
Installing Language Modules – 235
Overriding Existing Modules – 236
Codeless Language Modules – 236
Code-based Language Modules – 236
Language Module Compatibility – 236
Plug-In Obsolescence 237

Language Modules

Language modules are add-on items which provide syntax coloring and function browsing for programming languages that TextWrangler does not natively support.

There are two types of language modules: coded, and codeless. Coded language modules must be prepared according to the requirements of BBEdit's language module interface. (See Appendix C.) Codeless language modules are text documents prepared in a specific plist format. (See below.)

After you install a language module and relaunch TextWrangler, syntax coloring and function browsing will be available for the language(s) supported by that module. To verify that a language module is active, or to modify or add file suffix mappings for the language(s) it provides, use the Languages preference panel (see page 186).

Installing Language Modules

To install a language module, move or copy the module file into the Language Modules folder of your TextWrangler application support folder. If no such folder exists, you may create it.

After installing a new language module, you must quit and relaunch TextWrangler in order to use it.

To remove an installed language module, you must remove the item's file from the Language Modules subfolder of your TextWrangler application support folder, then quit and relaunch TextWrangler.

Overriding Existing Modules

Language modules can override existing language definitions, including the built-in definitions. If there is more than one module present which supports a given language, TextWrangler will use the module with the most recent modification date.

Codeless Language Modules

A codeless language module is a specially-formatted text file which allows you to describe the properties of a source code language via a set of basic parameters. TextWrangler will then use these parameters to perform syntax coloring and function navigation for the specified language.

Codeless language modules are written as "property lists" (or "plists"), which is anXML format that Mac OS X uses for many purposes. You can create or edit codeless language module files with TextWrangler itself, with the Mac OS X Property List Editor (located in /Developer/Applications/Utilities/if you have installed the Apple Developer Tools package), or with a third-party editor such as PlistEdit Pro.

`http://www.fatcatsoftware.com/plisteditpro/`

You can find complete specifications for creating codeless language modules in the Developer Information section of our web site.

`http://www.barebones.com/support/develop/`

Code-based Language Modules

TextWrangler also supports producing code-based language modules to handle more complex languages or document formats. You can find complete specifications for creating code-based language modules in the Developer Information section of our web site.

`http://www.barebones.com/support/develop/`

Language Module Compatibility

IMPORTANT You will **not** be able to use any third-party language modules which do not support Unicode text, or which were built in CFM format. If TextWrangler encounters such a module, it will not load that module, and will log a message to the system console.

Contact the developers of such a module, or visit the Bare Bones Software web site (see above) for more information on the availability of updated modules.

Plug-In Obsolescence

IMPORTANT TextWrangler 4.0 and later no longer support BBXT-based code plug-ins, and will not load any such items present in the “Plug-Ins” application support folder.

If you used any third-party commercial plug-in, please contact its developer for information on alternative solutions.

APPENDIX

A

Command Reference

This appendix provides a quick reference for key assignments and a comprehensive list of the commands that are available from TextWrangler's user interface.

In this appendix

Keyboard Shortcuts for Commands 239
Assigning Keys to Menu Commands 240
Available Key Combinations – 240
Listing by Menu and Command Name 241
Listing by Default Key Equivalent 246

Keyboard Shortcuts for Commands

Many of TextWrangler's commands have pre-defined keyboard shortcuts. TextWrangler also lets you reassign the shortcuts for any menu command, script, or text filter to suit your own way of working.

To change the keyboard shortcut for any menu command, you can use the Menus & Shortcuts preference panel. (See "Assigning Keys to Menu Commands" on the following page.)

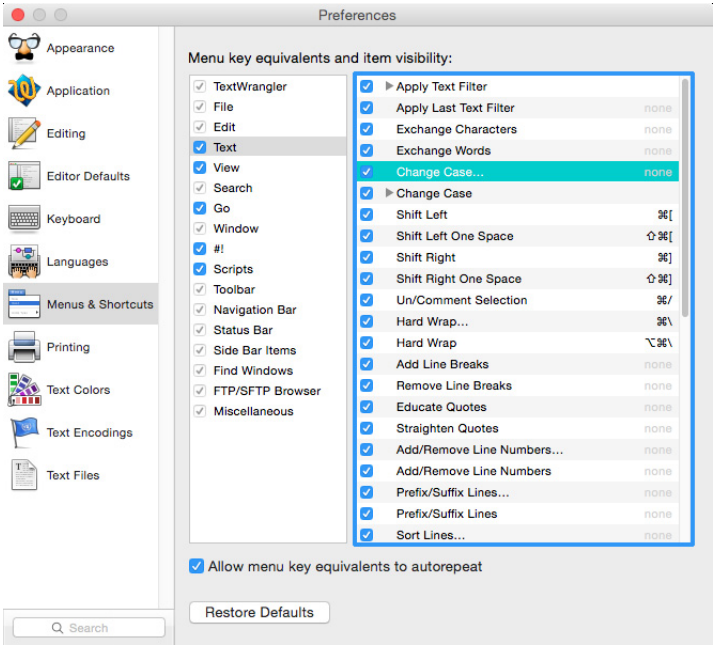
Many other TextWrangler features can have keyboard shortcuts assigned as well. Here's how to set them:

Feature	Set Keys in...
Menu commands	Menus & Shortcuts preference panel
Text filters	Menus & Shortcuts preference panel, or the Text Filters palette
Scripts	Menus & Shortcuts preference panel, or the Scripts palette
Stationery	Menus & Shortcuts preference panel

To display any of TextWrangler's floating palette windows, use the Palettes submenu in the Window menu.

Assigning Keys to Menu Commands

You can assign your own keyboard shortcuts (key equivalents) to any of TextWrangler's menu commands, as well as items on the Text Options, Markers, and Line Breaks toolbar popup menus, by choosing Preferences from the TextWrangler menu to bring up the Preferences window, then selecting the Menus & Shortcuts preference panel.



To set the key equivalent for a menu command, locate and select the entry for the command under the appropriate menu section, then double-click on the right-hand part of the line containing that command, and type the desired keystroke.

To remove an existing key equivalent from a command, double-click on the existing key combination and press the Delete key.

Click the Restore Defaults button to restore all key equivalents to their default values (as listed in this Appendix).

Available Key Combinations

All menu key combinations must include either the Command key or the Control key (or both), except function keys, which may be used unmodified. The Help, Home, End, Page Up and Page Down keys can be used in menu key combinations as well. The Help key can be assigned without modifiers; the others must be used in combination with at least either the Command or Control key.

Listing by Menu and Command Name

TextWrangler Menu

About TextWrangler	
Register...	
Preferences	Cmd-,
Setup	
Upgrade to BBEdit...	*(not in Mac App Store version)*
Check for Updates...	
Install Command Line Tools...	*(not in Mac App Store version)*
Services	*(submenu)*
Hide TextWrangler	Cmd-H
Hide Others/Show All	Cmd-Opt-H
Quit TextWrangler	Cmd-Q

File

New	*(see next page)*
New With Stationery	*(submenu)*
Open…	Cmd-O
Open from FTP/SFTP Server...	Cmd-Ctl-O
Open File by Name...	Cmd-D
Reveal Selection	Cmd-Opt-D
Open Counterpart	Cmd-Opt-uparrow
Open Recent	*(submenu)*
Reopen Using Encoding	*(submenu)*
Close Window	Cmd-Shift-W
Close All Windows	Cmd-Opt-W
Close Document	Cmd-W
Close All Documents	Cmd-Opt-Shift-W
Close & Delete	
Save	Cmd-S
Save All	Cmd-Opt-S
Save As…	Cmd-Shift-S
Save a Copy…	
Save to FTP/SFTP Server…	Cmd-Ctl-S
Save a Copy to FTP Server…	Cmd-Opt-Ctl-S
Revert	
Reload from Disk	
Export	
Hex Dump File	
Hex Dump Front Document..	
Page Setup…	
Print…	Cmd-P
Print All	Cmd-Opt-P
Print One Copy	Cmd-Shift-Opt-P
Print Selection	

Edit

Undo <action>	Cmd-Z
Redo <action>	Cmd-Shift-Z
Clear Undo History	Cmd-Ctl-Z
Cut	Cmd-X
Cut & Append	Cmd-Shift-X
Copy	Cmd-C
Copy & Append	Cmd-Shift-C
Paste	Cmd-V
Paste & Select	Cmd-Opt-V
Paste Previous Clipboard	Cmd-Shift-V
Paste Column	Cmd-Ctl-V
Clear	
New Line Before Paragraph	Cmd-Shift-Return
New Line After Paragraph	Cmd-Return
Move Line Up	Ctl-Up arrow
Move Line Down	Ctl-Dn arrow
Delete Line	Ctl-Shift-Delete
Select All	Cmd-A
Select None	Cmd-Shift-A
Select Line	Cmd-L
Select Paragraph	Cmd-Opt-L
Select Up	Ctl-Shift-Up arrow
Select Down	Ctl-Shift-Dn arrow
Insert	*(see next page)*
Copy Path	*(submenu)*
Show Clipboard	
Previous Clipboard	Ctl-[
Next Clipboard	Ctl-]
Text Options…	Cmd-Opt-,
Document Options…	Cmd-Ctl-,
Printing Options…	Cmd-Shift-,
Normalize Options…	
Special Characters…	

File --> New

Text Document	Cmd-N
(with selection)	
(with Clipboard)	
Text Window	Cmd-Shift-N
Disk Browser	Cmd-Opt-N
FTP/SFTP Browser	

Edit --> Insert

File Contents...
File/Folder Paths...
Folder Listing...
Page Break
Short Time Stamp
Full Time Stamp
Emacs Variable Block...

Edit --> Copy Path

Copy Path
Copy Full Path
Copy URL
Copy Name

Text

Apply Text Filter	*(submenu)*
Apply Text Filter <last filter>	
Exchange Characters	
Exchange Words	(Opt)
Change Case…	
Change Case	*(submenu)*
Shift Left	Cmd-[
Shift Left One Space	Cmd-Shift-[
Shift Right	Cmd-]
Shift Right One Space	Cmd-Shift-]
Un/Comment Selection	Cmd-/
Hard Wrap…	Cmd-\
Hard Wrap	Cmd-Opt-\
Add Line Breaks	
Remove Line Breaks	
Educate Quotes	
Straighten Quotes	
Add/Remove Line Numbers...	
Add/Remove Line Numbers	(Opt)
Prefix/Suffix Lines…	
Prefix/Suffix Lines	(Opt)
Sort Lines…	
Sort Lines	(Opt)
Process Duplicate Lines…	
Process Duplicate Lines	(Opt)
Process Lines Containing…	
Process Lines Containing	(Opt)
Increase Quote Level	
Decrease Quote Level	
Strip Quotes	
Zap Gremlins…	
Zap Gremlins	(Opt)
Entab…	
Entab	(Opt)
Detab…	
Detab	(Opt)
Normalize Line Endings	
Find Next Misspelled Word	Cmd-;
Find All Misspelled Words	Cmd-Opt-;
Clear Spelling Errors	
Check Spelling as You Type	
Show/Hide Spelling Panel	Cmd-Shift-;

View

Text Display	*(submenu)*
Show/Hide Toolbar	
Show/Hide Navigation Bar	
Show/Hide Editor	
Show/Hide Files	Cmd-0
Show/Hide Open Documents	
Show/Hide Recent Documents	
Show/Hide Worksheet & Scratchpad	
Balance	Cmd-B
Balance & Fold	Cmd-Shift-B
Fold Selection	
Unfold Selection	
Collapse Enclosing Folds	
Collapse All Folds	
Expand All Folds	
Previous Document	Cmd-Opt-[
Next Document	Cmd-Opt-]
Move to New Window	Cmd-Opt-O
Open in Additional Window	
Reveal in Finder	
Go Here in Terminal	
Go Here in Disk Browser	
Enter Full Screen	

View -> Text Display

Show/Hide Fonts	Cmd-T
Soft Wrap Text	
Show/Hide Page Guide	
Show/Hide Tab Stops	
Show/Hide Line Numbers	
Show/Hide Gutter	
Show/Hide Invisibles	
Show/Hide Spaces	

Search

Find…	Cmd-F
Multi-File Search	Cmd-Shift-F
Search in Document's Folder	
Search in Disk Browser	
Live Search	Cmd-Opt-F
Find Next	Cmd-G
Find Previous	Cmd-Shift-G
Find All	Cmd-Opt-G
Find Selected Text	(Cmd-H *or none*)
Find Previous Selected Text	Cmd-Shift-H
Use Selection for Find	Cmd-E
Use Selection for Find (grep)	Cmd-Shift-E
Use Selection for Replace	Cmd-Opt-E
Use Selection for Replace (grep)	Cmd-Opt-Shift-E
Replace	Cmd-=
Replace All	Cmd-Opt-=
Replace All in Selection	Cmd-Ctl-=
Replace to End	Cmd-Shift-=
Replace & Find Next	
Find Differences…	
Compare Two Front Windows	
Compare Against Disk File	
Apply to New	Cmd–left arrow
Apply to Old	Cmd–right arrow
Compare Again	
Find in Reference	Cmd-Shift-hyphen

Go

Line Number…	Cmd-J
Line Number	Cmd-Opt-J
Center Line	Cmd-Shift-J
Functions	
Reveal Start	
Reveal End	
Go to Previous	
Go to Next	
Jump Points	
Previous	
Next	
Set	

Window

Minimize Window	
Minimize All Windows	(Opt)
Bring All to Front	
Palettes	(*submenu)*
Save Default [type of] Window	
Cascade Windows	
Tile Two Front Windows	(Option)
Arrange	(*submenu)*
Tile Two Front Windows	(Opt)
Cycle Through Windows	Cmd-`
Cycle Through Windows Backwards	Cmd-Shift-`
Exchange With Next	
Synchro Scrolling	
(Open windows)	

Window -> Palettes

Character Inspector
Colors
Scripts
Text Filters
Windows

Shebang (#!)

Check Syntax	
Check Selection Syntax	(Opt)
Run…	
Run	
Run in Terminal	
Run in Debugger	
Run File…	
Show Module Documentation	

Scripts

Open Script Editor
Open Scripting Dictionary
Open Scripts Folder
Start/Stop Recording
(Installed scripts)

Help

Search	(menu command search field)
TextWrangler Help	
User Manual	
Tutorial	
Service and Support	

Toolbar *(in-window)*

Text Options	(popup menu)
Soft Wrap Text	
Show/Hide Page Guide	
Show/Hide Tab Stops	
Show/Hide Line Numbers	
Show/Hide Gutter	
Show/Hide Invisibles	
Show/Hide Spaces	
Use Typographer's Quotes	
Auto-Expand Tabs	

Navigation Bar *(in-window)*

Open Files Menu	Ctl-Opt-F
Open Function Menu	Ctl-Opt-N
Open Includes Menu	Ctl-Opt-I
Open Marker Menu	Ctl-Opt-M
Markers	(popup menu)
Set Marker…	
Set Marker	(Opt)
Clear Markers…	
Clear All Markers	(Opt)
Find & Mark All…	
Find & Mark All	(Opt)

Status Bar *(in-window)*

Open Language Menu	
Open Text Encodings Menu	
Open Breaks Menu	Ctl-Opt-B
Line Breaks	(popup menu)
Classic Mac (CR)	
Unix (LF)	
Windows (CRLF)	

Miscellaneous Commands

Zoom Window	
Zoom All Windows	
Zoom Window Full Screen	
Zoom All Windows Full Screen	
Open URL	(Cmd-click within any URL)

Listing by Default Key Equivalent

Key	Command
Cmd-0	View: Show/Hide Files
Cmd-A	Edit: Select All
Cmd-B	Text: Balance
Cmd-C	Edit: Copy
Cmd-D	File: Open Selection *or* File: Open File by Name
Cmd-E	Search: Use Selection for Find
Cmd-F	Search: Find…
Cmd-G	Search: Find Next
Cmd-H	TextWrangler: Hide TextWrangler
Cmd-J	Go: Line Number…
Cmd-L	Edit: Select Line
Cmd-N	File: New: Text Document
Cmd-O	File: Open…
Cmd-P	File: Print…
Cmd-Q	TextWrangler: Quit TextWrangler
Cmd-S	File: Save
Cmd-T	View: Text Display: Show/Hide Fonts
Cmd-V	Edit: Paste
Cmd-W	File: Close Document/Close Window
Cmd-X	Edit: Cut
Cmd-Z	Edit: Undo
Cmd-,	TextWrangler: Preferences
Cmd-`	Window: Cycle Through Windows
Cmd-;	Text: Find Next Misspelled Word
Cmd-[	Text: Shift Left
Cmd-]	Text: Shift Right
Cmd-/	Un/Comment Selection
Cmd-=	Search: Replace

Key	Command
Cmd-\	Text: Hard Wrap…
Cmd–left arrow	Search: Apply to New
Cmd–right arrow	Search: Apply to Old
Cmd-Ctl-O	File: Open from FTP/SFTP Server…
Cmd-Ctl-S	File: Save to FTP/SFTP Server…
Cmd-Ctl-V	Edit: Paste Column
Cmd-Ctl-Z	Edit: Clear Undo History
Cmd-Opt-D	File: Reveal Selection
Cmd-Opt-E	Search: Use Selection for Replace (grep)
Cmd-Opt-F	Search: Live Search
Cmd-Opt-G	Search: Find All
Cmd-Opt-H	TextWrangler: Hide Others
Cmd-Opt-L	Edit: Select Paragraph
Cmd-Opt-N	File: New: Disk Browser
Cmd-Opt-O	View: Move to New Window
Cmd-Opt-P	File: Print All
Cmd-Opt-S	File: Save All
Cmd-Opt-V	Edit: Paste & Select
Cmd-Opt-W	File: Close All Windows
Cmd-Opt-,	Edit: Text Options
Cmd-Opt-;	Edit: Find All Misspelled Words
Cmd-Opt-[	View: Previous Document
Cmd-Opt-]	View: Next Document
Cmd-Opt-=	Search: Replace All
Cmd-Opt-\	Text: Hard Wrap
Cmd-Opt–up arrow	File: Open Counterpart

Key	Command
Cmd-Opt-Shift-E	Search: Use Selection for Replace (grep)
Cmd-Opt-Shift-P	File: Print One Copy
Cmd-Opt-Shift-W	File: Close All Documents
Cmd-Shift-A	Edit: Select None
Cmd-Shift-B	View: Balance & Fold
Cmd-Shift-C	Edit: Copy & Append
Cmd-Shift-E	Search: Use Selection for Find (grep)
Cmd-Shift-F	Search: Multi-File Search
Cmd-Shift-G	Search: Find Previous
Cmd-Shift-J	Search: Go to Center Line
Cmd-Shift-N	File: New: Text Window
Cmd-Shift-P	File: Page Setup
Cmd-Shift-S	File: Save As…
Cmd-Shift-V	Edit: Paste Previous Clipboard
Cmd-Shift-W	File: Close Window *{special}*
Cmd-Shift-X	Edit: Cut & Append
Cmd-Shift-Z	Edit: Redo
Cmd-Shift-,	Edit: Printing Options
Cmd-Shift-`	Misc.: Cycle Through Windows Backwards
Cmd-Shift--	Search: Find in Reference
Cmd-Shift-;	Text: Show Spelling Panel
Cmd-Shift-[	Text: Shift Left One Space
Cmd-Shift-]	Text: Shift Right One Space
Cmd-Shift-Return	New Line Before Paragraph
Ctl-[	Edit: Previous Clipboard
Ctl-]	Edit: Next Clipboard
Ctl-Tab	(switch focus to next pane)
Ctl-Down arrow	Edit: Move Line Down

Key	Command
Ctl-Up arrow	Edit: Move Line Up
Ctl-Opt-C	Navigation Bar: Open Counterpart Menu
Ctl-Opt-F	Navigation Bar: Open Files Menu
Ctl-Opt-I	Navigation Bar: Open Includes Menu
Ctl-Opt-M	Navigation Bar: Open Marker Menu
Ctl-Opt-N	Navigation Bar: Open Function Menu
Ctl-Shift-Delete	Edit: Delete Line
Ctl-Shift-Dn arrow	Edit: Select Down
Ctl-Shift-Up arrow	Edit: Select Up

APPENDIX

B

Editing Shortcuts

In TextWrangler you can perform many editing functions (including word selection or deletion) directly from the keyboard. Chapter 4 contains complete details on TextWrangler's text editing features. This appendix provides a quick reference to available keyboard and mouse shortcuts for word selection and deletion.

In this appendix

Mouse Commands 251
Arrow and Delete Keys 252
Emacs Key Bindings 253
Using universal-argument – 254

Mouse Commands

	No Modifier	Shift
Click	move insertion point	extend selection
Double-click	select word	extend selection to word
Triple-click	select line	–none–

Triple-clicking is the same as clicking in a line and then choosing the Select Line command from the Edit menu.

Holding down the Command or Option keys as you click or double-click triggers special actions:

	Option	Command	Command/ Option
Click	–none–	Open URL	–none–
Double-click	–none–	–none–	find next instance of the selected text

Arrow and Delete Keys

You can use the arrow keys to move the insertion point right, left, up, and down. You can augment these with the Command and Option keys to move by word, line, or screens, or with the Shift key to create or extend selections. For example, pressing Shift-Option-Right Arrow selects the word to the right of the insertion point.

You can hold down the Control key while using the arrow keys to scroll through editing windows without moving the position of the insertion point.

Key	Modifier	Action
(left/right) Arrow		Move 1 character left/right
(left/right) Arrow	Option	Move 1 word left/right
(left/right) Arrow	Command	Move to beginning/end of line
(left/right) Arrow	Control	Jump to the previous/next character transition from lower case to upper case OR the next word boundary
(up/down) Arrow		Move up/down 1 line in file
(up/down) Arrow	Command	Move to top/bottom of file
(up/down) Arrow	Option	Move to previous/next screen page
(up/down) Arrow	Control	Scroll view up/down
[any of the above]	Shift	Make or extend a selection range
Delete		Deletes selection range, or character preceding (to the left of) the insertion point.
Delete	Command	Deletes all characters backwards to beginning of line
Delete	Option	Deletes all characters back to beginning of word
Delete	Shift	(same as Forward Delete)
Forward Delete		Deletes selection range, or character after (to the right of) the insertion point
Forward Delete	Command	Deletes all characters forward to end of the current line
Forward Delete	Option	Deletes all characters forward to end of word
Forward Delete	Shift	(same as Forward Delete alone)

Note The meaning of the Command and Option modifiers listed above may be exchanged, depending on which settings you have selected for Exchange Command and Option Key Behavior in the Keyboard preference panel .

Emacs Key Bindings

The Keyboard preference panel contains an option labelled Use Emacs Key Bindings. When this option is on, TextWrangler will accept the following Emacs-style keyboard navigation commands. The Escape key is used in lieu of the Emacs “Meta” key; to type these key equivalents, press and release the Escape key followed by the specified letter key—for example, to type “Esc-V” press and release the Escape key and then type the letter V.

Key Sequence	Action
Ctl-A	beginning-of-line (Move insertion point to start of current line)
Ctl-B	backward-char (Move insertion point backward 1 place)
Ctl-D	delete-char (Delete forward 1 character)
Ctl-E	end-of-line (Move insertion point to end of current line)
Ctl-F	forward-char (Move insertion point forward 1 place)
Ctl-G	keyboard-quit (cancel pending arguments)
Ctl-K	kill-line (Cut)
Ctl-L	recenter (Scrolls the current view so the selection is centered on screen)
Ctl-N	next-line (Move insertion point down one line)
Ctl-O	open-line (Inserts line break without moving insertion point)
Ctl-P	previous-line (Move insertion point up one line)
Ctl-R	isearch-backward (Live Search backward)
Ctl-S	isearch-forward (Live Search forward)
Ctl-T	transpose-chars (Exchange Characters)
Ctl-U	universal-argument (*See note below*)
Ctl-V	scroll-up (Page down)
Ctl-W	kill-region (Cut)
Ctl-Y	yank (Paste)
Ctl-_	undo (Undo)

Key Sequence	Action
Ctl-X Ctl-C	save-buffers-kill-emacs (Quit)
Ctl-X Ctl-F	find-file (Open file)
Ctl-X Ctl-S	save-buffer (Save current document)
Ctl-X Ctl-W	write-file (Save As)
Esc-<	beginning-of-buffer (Move insertion point to start of document)
Esc->	end-of-buffer (Move insertion point to end of document)
Esc-Q	fill-paragraph (Hard Wrap with current settings)
Esc-T	transpose-words (Exchange Words)
Esc-V	scroll-down (Page up)
Esc-W	copy-region-as-kill (Copy)
Esc-Y	yank-pop (Paste Previous Clipboard)

Using universal-argument

The universal-argument command (Ctl-U) does not work quite the same way as it does in Emacs. In TextWrangler, it is a simple repeat-count. For example, if you type Ctl-U, then a 3, and then Ctl-N, the insertion point will move down three lines. There is no visual feedback as you type the number, and no way to backspace or otherwise edit the number. If you make a mistake, the best you can do is type Ctl-G (keyboard-quit) and start over.

APPENDIX

C Codeless Language Modules

The information previously contained in this appendix is now superseded by the Codeless Language Module Reference on our website:

`http://www.barebones.com/support/develop/clm.html`

Index

Symbols

"Home" and "End" Keys 185

A

active windows 60
alternation 148
AppleScript 30, 34
    attaching scripts to menu items 210
    pitfalls 224
    reading dictionary 218
    recording 210
application launch
    overriding default action 181
application launch behavior 181
Application Preferences 180
Apply to New command 133
Apply to Old command 133
Arrange command 115
arranging windows 115
arrow keys 252
ASCII table 114
attaching scripts to menu items 210
automatic spell checking 184

B

backups 56
bbdiff 95
binary plist files 44
BOM. see byte-order mark 41, 49
Bonjour 51
bookmarks 51
browsers 169
    differences 95
    disk browser 171
    file list panel 172
    search results 123, 173
    setting the list display font 180
    splitter 170
    status bar 170, 171
    text panel 170
byte-order mark 41, 49
byte-swapped. see Little-Endian 49
bz2-compressed files 44

C

C programming language 111
camel case. see CamelCase 79
CamelCase 79
    keyboard navigation of 79
Cancel button 22
capitalize
    lines 105
    sentences 105
    words 105
Cascade Windows command 115
Cascade Windows, see also Arrange 115
case sensitivity 120
case transformations 152
changing case 104
character classes 142
character offset specification 45
character set encoding 37, 41, 49
check spelling as you type 184
Check Spelling command 100
checking spelling
    user dictionary 102
Classic Mac line breaks 37
Clear command 22, 60
Clear key 60
clearing a marker 99
Clipboard 61
clipboard 61
clipboards, multiple 61
colored text 87
columns
    see rectangular selection 79
command keys
    assigning to menu items 239
    in dialogs 22
    in menus 22
    listing by default key 246
    listing by menu 241
    shortcuts 251
command line
    configuring TextWrangler for use with 225
Command-Period 22
Compare Again command 133
Compare Against Disk File 96
Compare Two Front Windows 94
comparing files 94
    multiple files 97
complex patterns 146
concatenate 93
contextual menu, in disk browsers 172
control characters 111
Convert to ASCII 106

Copy & Append command 61
Copy command 22, 61
Counterpart button 68
counterparts
    overriding defaults 47
creating documents 35
    with clipboard 35
    with selection 35
cursor movement 78
    using arrow keys 79
Cut & Append command 61
cut and paste 60
Cut command 22, 60

D

default window position
    setting 115
Delete key 60, 83, 252
deleting text 60
Detab command 112
dialog keyboard shortcuts 22
dictionary, AppleScript 218
Differences command 95
disclosure triangles 73
Disk Browser 35
disk browsers 30, 34, 35, 40, 171
    file list panel 172
    status bar 171
document proxy icon 64, 77
documents
    comparing 94
    creating 35
    editing text 60
    inserting text 93
    opening 257
    saving 35, 36
    window anatomy 63
    window handling 257
documents drawer. see file list 69
DOS line breaks
    see Windows line breaks 37
double-clicking 40
drag-and-drop
    in document windows 61
    to TextWrangler application icon 40
    to Windows floating window 40
drawer. see file list 69
dynamic menus 21

E

edit tool 226
editing text 60
    shortcuts 251
Editor Defaults 179
Emacs Key Bindings 186, 253
Emacs variables 39
    x-counterpart 47
encoding 37, 41, 49
End key 84
Entab command 112
Enter key 22
escape codes 139
Escape key 22
Exchange with Next command 115
expanding tabs 86
extending the selection 79, 84

F

F keys 84
Favorites 32
file filters 126
file groups 35
file list 69
file list panel 172
file transfer format, FTP/SFTP 53
Filters 233
filters, file 126
Find & Mark All command 99
Find & Replace All Matches 129
Find Again command 119, 131
Find All 119, 123
Find command 117, 120, 130
Find dialog
    see Find window 118
Find Differences command 132
Find Selection command 131
Find window 118
finding text
    see searching
floating windows
    ASCII table 114
    window list 114
fold indicator 74, 76
folder, listing contents of 93
foreign text 103
Forward Delete key 83, 84
Frame Printing Area 57
freezing line endings 89
FTP
    alternate ports 53
FTP Browsers 55
function keys 84
function navigation. see function popup 66
function popup 66

G

Go To Center Line command 134
Go To Line command 84, 134

gremlins 110
grep 120
    alternation 148
    backreferences 156
    character classes 142
    comments 159
    complex patterns 146
    conditional subpatterns 163
    entire matched pattern 150
    escape codes 139, 143
    examples 153
    excluding characters 142
    longest match issue 148
    lookahead assertions 162
    lookbehind assertions 162
    marking a mail digest 155
    marking structured text 154
    matching delimited strings 154
    matching nulls 156
    matching white space 153
    matching words and identifiers 153
    named backreferences 147
    named subpattern 146
    non-capturing parentheses 158
    non-printing characters 143
    non-repeating subpatterns 164
    numbered backreferences 147
    once-only subpatterns 164
    pattern modifiers 160
    positional assertions 161
    POSIX character classes 157
    quantifiers 145
    ranges 142
    rearranging name lists 155
    recursive patterns 166
    repetition 145
    replacement patterns 150
    replacing with subpatterns 151
    setting markers with 99
    subpatterns 146, 150
    wildcards 140
Grep Patterns.xml 30
gzip-compressed files 44

H

Hard Wrap command 89, 91
hard wrapping 88, 90, 105
headers 58
hex escapes 121, 143
hexadecimal 111
hidden files
    on FTP servers 52
Highlight Insertion Point 191
highlighting of text 60
Home key 84
human interface 21

I

incremental search
    see Live Search 130
indenting 105
input, Unix filter 233
inserting files 93
inserting folder listings 93
inserting page breaks 93
inserting text 93
insertion point 60
Install Command Line Tools 55, 226
installing TextWrangler 25
international text 37, 41, 49, 103
invisible characters 87
invisible files 44
    on FTP servers 52

K

Key Equivalent 246
keyboard shortcuts 240, 251
    in dialogs 22

L

language, source code 86
launching TextWrangler 34
line breaks 37, 105
line breaks, default 37
line ending format 37
line endings 37
Line Number command 134
line number specification 45
line numbers
    show on printout 57
list display font 180
Little-Endian 49
Live Search 130
longest match issue 148
lower case 105

M

Mac line breaks
    see Classic Mac 37
Macintosh Drag and Drop 61
    see also drag-and-drop
Mark pop-up menu 98
Marker popup menu 68
markers
    clearing 99
    setting 99
menus 21, 22
Menus & Shortcuts preference panel 22

Menus preference panel 239
mouse shortcuts 251
moving text 60
moving the cursor 78
    using the arrow keys 79
multi-byte text 37, 41, 49, 103
multi-file comparisons 97
multi-file search 121
Multi-File Search command 117
multiple clipboards 61
multiple Undo 62

N

named subpattern 146
navigation
    functions 66
navigation bar 65
nested folds 73
New Window with Selection 35
Non-Greedy Quantifiers 149
non-printing characters 87, 120
numeric keypad 83

O

Open command 40
    options 42
Open dialog 44
Open File by Name command 48
Open from FTP/SFTP Server 44
    Show Files Starting with "." 52
Open Hidden
    see Show Hidden Items 44
Open Recent command 40, 47
Open Recent menu 180
Open Selection 45
Open Selection command 40, 44
Opening 40
    binary plists 44
    bz2-compressed files 44
    gzip-compressed files 44
Opening Existing Documents 40
Option-¥ on Japanese Keyboards 186
outdenting 105

P

page breaks 93
Page Down key 85
Page Guide 184
page guide 86
Page Up key 85
paragraph (definition) 60
Paragraph Fill option 91
Paste command 22, 61
Paste Previous Clipboard 254
Paste Previous Clipboard command 61
pattern matching
    see grep
Perl 229
Perl scripts 229
Perl/Unix Filters palette 239
POSIX-Style Character Classes 157
Preferences 175
    Application 180
    Function Popup 226
    Printing 57
Prefix/Suffix Lines plug-in 107
Print Color Syntax 58
Print Line Numbers 57
Print One Copy command 57
Print Selection 57
printing 57
Process Lines Containing plug-in 109
pull-down menus 21
Python 229
    configuration 231
Python scripts 229

Q

Quick Search
    see Live Search 130

R

range end indicators 74
recording scripts 210
rectangular selection 79
Redo command 62
reflowing paragraphs 91
refresh open files 180
regular expressions
    see grep
remember recently used items 180
Remove Line Breaks command 89
Rendezvous. see Bonjour 51
Reopen Documents 181
Reopen documents, preventing 181
repetition metacharacters 145
Replace 119
Replace & Find Again command 120, 132
Replace All 119, 123, 129, 132
Replace to End 120
replacing text 60
    see also searching
Return key 22
Ruby 229

S

Save a Copy command 36
Save a Copy to FTP Server command 54

Save As command 36
Save As options
line breaks 37
Save command 36
Save Selection command 36
Save to FTP Server command 53
Saved Sources.xml 32
script systems 103
Scripts 234
Scripts folder 29
Scripts palette 29, 239
scrolling, synchronized 115
search results window 123, 173
search sources 32
searching 118
all open documents 125
case sensitive 120
for non-printing characters 120
for whole words 120
grep 120
see also grep
in a folder 125
in multiple files 121
in results of a previous search 125
in selection only 120
menu reference 130
non-printing characters 143
replacing in multiple files 129
results window 123, 173
search set 124
wrap around 120
Select All command 22, 60
Select Line command 60
Select Paragraph command 60
selected text 60
selecting text 60, 78
by clicking 78
extending the selection 79
rectangular selection 79
Services menu 35
Set (jump mark) 135
Set Marker command 99
Set Menu Keys. see Menus & Shortcuts preference panel 22
setting markers 99
using grep 99
Shell scripts 229
shell scripts 229
shell worksheets 226
shifting text 105
Show Files Starting with "." 52
Show Hidden Items 44
Show Invisibles command 64
Show Page Guide 64, 179
Soft Wrap Text command 64
soft wrapping 86, 88, 89
as default 89
Software Update 180
Sort Lines plug-in 107
spell checking 100
user dictionary 102
split bar 65
in browsers 170
splitting a window. see split bar 65
startup
window handling 181
startup items 34
stationery 55
creating 55
Stationery folder 31
using 55
status bar
hiding 87
in browsers 170
in disk browsers 171
status bar. see tool bar 63
subpatterns 146
Synchro Scrolling command 115
syntax coloring 87
on printout 58

T

tab width 179
tabs
converting to and from spaces 112
tarballs 44
Text Display menu 75
Text Document, creating 35
text encoding
choosing 41
Text Encodings preference panel 41
text folding
disclosure triangles 73
fold indicator 74
nested folds 73
range end indicators 74
text highlighting 60
Text Options popup 64
text transformation 88
text wrapping 88
TextWrangler Talk discussion group 167
time stamps 58
tool bar 63
transformations, case 152
twdiff tool 227
twfind tool 227
typing text 60
typographer’s quotes 86

U

Un/Comment command 105
Undo command 62
Unicode 37, 41, 49, 103
universal-argument 254
Unix line breaks 37
Unix shell scripts 229
user interface 21
Using Language Modules 236
UTF-16 41, 49
UTF-8 41, 49

V

verify open files 180

W

wildcards 140
window list 114
windows
    arranging 115
    exchanging with next 115
    split bar 65
Windows floating window 40
Windows line breaks 37
Windows menu 113
worksheets, shell 226
wrap around 120
wrapping text 86, 88

Y

yank-pop 254

Z

Zap Gremlins command 110